高等职业教育“十二五”规划教材

手机应用软件开发与应用项目实践（Android版）

刘 旭 万春旭 ◎ 主 编
张红玉 尹 娜 ◎ 副主编

- PPT教学课件
- 游戏源代码
- 两款游戏的全程开发详解
- 理论与实践紧密结合

配套资源下载地址：http://www.tup.com.cn

清华大学出版社

高等职业教育“十二五”规划教材

手机应用软件开发与应用项目实践（Android版）

刘　旭　万春旭　主　编

张红玉　尹　娜　副主编

清华大学出版社

北　京

内 容 简 介

本书根据高职高专层次的教学大纲要求，从方便读者理解且易于上手的角度出发，把全书分为两大篇：基础篇和高级篇。基础篇主要介绍了Android开发的基础知识，包括Android基本概念、OMS基本介绍、Android开发环境的搭建、开发第一个Android项目及Android程序的监控与调试等内容。高级篇详细介绍了两款不同类型的Android游戏的开发过程，如开发步骤和代码原理，以帮助读者掌握相关知识。另外，本书在每小节的最后都配有小知识，或是拓展本节的技术内容，或是介绍当前移动互联网产业的新发展，具有很大的启发性，在帮助读者理解知识点的同时了解产业动态。

本书内容系统、全面，讲解通俗易懂、难度适中。通过本书的学习，读者能够快速掌握Android游戏开发技巧，为日后自主开发打下坚实基础。

本书可作为高等职业院校、高等专科院校计算机相关专业的教材，也可作为非计算机专业学生选修课教材，还可作为计算机应用人员的自学参考书。

图书在版编目（CIP）数据

手机应用软件开发与应用项目实践（Android版）/刘旭、万春旭主编．—北京：清华大学出版社，2012.11
高等职业教育“十二五”规划教材

ISBN 978-7-302-29688-1

Ⅰ. ①手…　Ⅱ. ①刘…　②万…　Ⅲ. ①移动电话机-应用软件-高等职业教育-教材　Ⅳ. ①TN929.53

中国版本图书馆CIP数据核字（2012）第187372号

责任编辑：杜长清
封面设计：刘　超
版式设计：文森时代
责任校对：张彩凤
责任印制：
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　**印　　张：**15.75　**字　　数：**364千字
版　　次：2012年11月第1版　**印　　次：**2012年11月第1次印刷
印　　数：1～4000
定　　价：29.00元

产品编号：048626-01

丛书编委会

丛书编委会院校名单

（按拼音排序）

包头轻工职业技术学院
北京城市学院
北京农业职业学院
北京印刷学院
大连海洋大学职业技术学院
大连艺术学院
广东科技学院
广东省惠州市惠城区技工学校
广西工商职业技术学院
广西玉林师范学院
河北青年管理干部学院
河北省沙河市职教中心
河南工业职业技术学院
河南化工职业学院
河南中医学院信息技术学院
黑龙江农业工程职业学院
衡水职业技术学院
湖北文理学重庆教育学院
湖南省衡阳技师学院
湖南信息职业技术学院
华南师范大学
黄河水利职业技术学院
黄山学院信息工程学院
吉林电子信息职业技术学院
吉林省四平市四平职业大学
江苏经贸职业技术学院
军事经济学院襄樊分院
昆明工业职业技术学院
兰州外语职业学院
辽宁信息职业技术学院
聊城市高级技工学校
临汾职业技术学院
临沂职业学院
洛阳师范学院
吕梁学院
内蒙古机电职业技术学院
宁夏工商职业技术学院
青海畜牧兽医职业技术学院
厦门软件学院
山东省潍坊商业学校
山东师范大学
山东信息职业技术学院
山西青年职业学院
首钢工学院
四川大学锦江学院
四川职业技术学院
太原大学
泰山职业技术学院
唐山工业职业技术学院
天津青年职业学院
潍坊职业学院
武汉商业服务学院
烟台工程职业技术学院
扬州工业职业技术学院
张家口职业技术学院
郑州轻工业学院
郑州铁路职业技术学院
重庆教育学院
淄博职业学院
浙江商业职业技术学院

前　　言

一、本书特色

本书着重介绍了两款基于 Android 系统的手机应用游戏，目的在于通过实例帮助读者了解 Android 开发过程。本书特色有三：

一是实例恰当。战国英雄传和微博随身听是两款完全不同且具有代表性的游戏类型，读者在掌握了这两款游戏的开发后可基本掌握 Android 开发技能，方便进行其他相关项目的开发。

二是讲解详细。从 Android 开发、程序的建立到两款游戏的每个步骤，都有详细的讲解，十分有利于初学者学习，读者只需跟随书中的介绍一步步操作实践，即可完成游戏开发。

三是在介绍游戏开发的过程中插入了很多小知识，其内容不仅涉及所用知识点的详细介绍、扩展补充，还包括了产品设计其他环节的简单介绍，如构图、配色等，可以启发读者在产品设计其他环节的思考。另外还有一些是游戏设计从业者多年的心得、体会，可帮助读者更好、更全面地了解这一工作，甚至还有对移动互联网产业的最新分析，可使读者从更高的层面了解相关工作，拓展读者的行业视野。

二、如何用好本书

如前面所讲，本书十分适合初学者阅读，阅读时一定要跟随书中的实例一步步操作实践，切忌眼高手低。读者只需跟随书中的介绍将两款游戏开发完成，即可基本掌握 Android 的开发技能，尤其是 Android 手机游戏的开发技能。

三、作者寄语

希望读者能够通过书中实例的学习真正了解并掌握 Android 开发的基本技能，并对产品设计的完整环节有所掌握，有兴趣的读者可进行进一步的学习，在学习的过程中有任何问题可发邮件至 ducqing@163.com。

编　者

目　　录

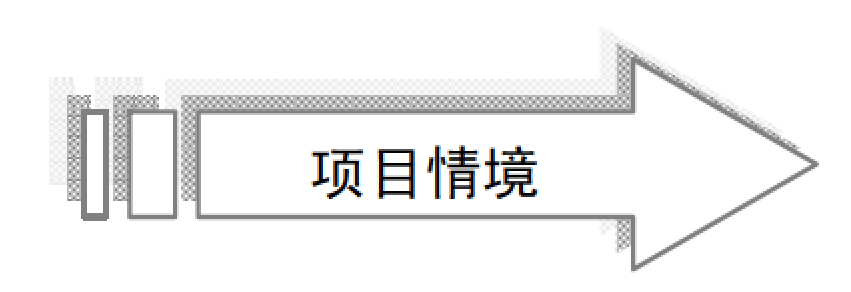
第一部分　基础篇——Android 基础

第二部分　高级篇——游戏综合实现

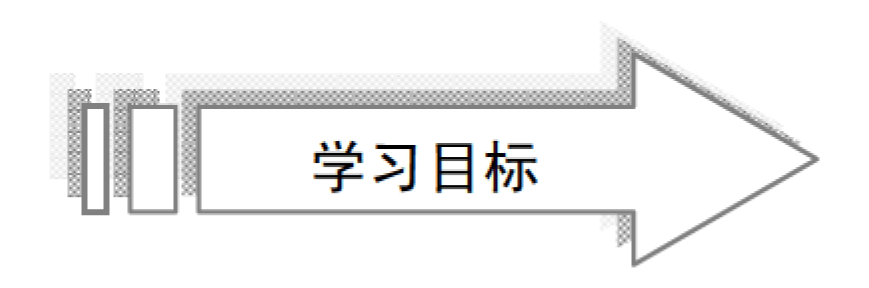
第一部分

基础篇——Android 基础

项目一
掀起 Android 的盖头



要进行 Android 开发，首先需要了解 Android 的基本知识，本项目将带领读者对 Android 的基本概念和相关知识进行相应的了解。



- 了解 Android 基本概念
- 了解 OMS 基本情况

任务 1　Android 基本概念

【任务情境】

Android 一词本意是指“机器人”，当然现在大家都知道它是 Google 推出的开源手机操作系统。Android 基于 Linux 平台，由操作系统、中间件、用户界面和应用软件组成，号称是首个为移动终端打造的真正开放和完整的移动软件。它是由 30 多家科技公司和手机公司组成的“开放手机联盟”共同研发的，这将大大降低新型手机设备的研发成本。完全整合的全移动功能性产品成为“开放手机联盟”的最终目标。

【相关知识】

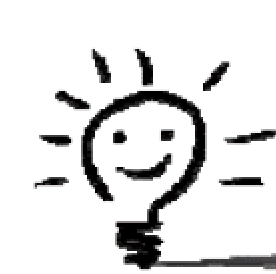
1. Android 简介

Android 作为 Google 移动互联网战略的重要组成部分，将进一步推进“随时随地为每个人提供信息”这一企业目标的实现。Google 的目标是让移动通信不依赖于设备，甚至是平台。出于这个目的，Android 将完善而不是替代 Google 长期以来推行的移动发展战略：通过与全球各地的手机制造商和移动运营商成为合作伙伴，开放既实用又有吸引力的移动服务，并推广这些产品。

Android 平台的研发团队阵容强大，包括 Google、HTC（宏达电）、T-Mobile、高通、摩托罗拉、三星、LG 以及中国移动在内的 30 多家企业都将基于该平台开发手机的新型业务，应用的通用性和互联性将在最大程度上得到保持。“开放手机联盟”表示，Android 平台可以促使移动设备的创新，让用户体验到最优质的移动服务。同时，开发商也将得到一个新的开放级别，更方便地进行协同作用，从而保障新型移动设备的研发速度。因此 Android 是第一个完整、开放、免费的手机平台。

Android 系统具有如下 5 个特点：

- ☑ 开放性。Google 与“开放手机联盟”合作开发了 Android，Google 通过与运营商、设备制造商、开发商和其他有关各方结成深层次的合作伙伴关系，希望通过建立标准化、开放式的移动电话软件平台，在移动产业内形成一个开放式的生态系统。
- ☑ 应用程序无界限。Android 上的应用程序可以通过标准 API 访问核心移动设备。通过互联网、应用程序声明它们的功能可供其他应用程序使用。
- ☑ 应用程序是在平等的条件下创建的。移动设备上的应用程序可以被替换或扩展，即使是拨号程序或主屏幕这样的核心组件。
- ☑ 应用程序可以轻松地嵌入网络。如可以轻松地嵌入 HTML、JavaScript 和样式表，还可以通过 Web View 显示网络内容。
- ☑ 应用程序可以并行运行。Android 是一种完整的多任务环境，应用程序可以在其中并行运行。在后台运行时，应用程序可以生成通知以引起注意。



为什么 Android 手机如此受用户青睐，下面来看看 Android 究竟有哪些功能在吸引着我们。

（1）智能虚拟键盘。虚拟键盘的出现意味着基于 Android 1.5 或以上版本（Android 2.0）的移动设备可以同样支持物理键盘和虚拟键盘。不同的输入方式可满足用户在特定场景的需求。Android 虚拟键盘可以在任何应用中提供，包括 Gmail、浏览器、SMS，当然也包括大量的第三方应用，如自动校正、推荐、用户词典等。不同于其他手机平台，Android 1.5 以上的版本还支持第三方虚拟键盘应用的安装。

（2）使用 Widget 实现桌面个性化。可以用 Widget“武装”自己的桌面。大多数小的 Web 应用都是从网络上获得实时数据并展示给用户的。Android 预装了 5 个桌面 Widget，包括数字时钟、日历、音乐播放器、相框和搜索。不同于 iPhone，Android 通过内置的应用程序库可安装第三方 Widget。

（3）用在线文件夹快速浏览在线数据。类似于 OS X Leopard 的 QuickLook 特征，Android 的在线文件夹可显示常见的数据条目，例如联系人、喜欢的应用、E-mail 信息、播放列表、书签、RSS 源等，并不需要运行系统程序处理特定的数据条目。在线文件夹数据会实时更新，就像通过云或是本地创建新的数据。开发者可以拓展通用数据条目和注册新数据类型的内置支持。例如，Twitter 客户端程序可以注册 tweet 作为新数据类型，因此可以从朋友那里创建 tweet 的在线文件。Android 可以为个人桌面提供一组在线文件夹，从而帮助我们快速、方便地浏览联系人、股市、书签等信息。

（4）视频录制和分享。Android 还有录制和分享视频的功能，对回放和 MPEG-4、3GP 等视频格式也有了更好的支持。可以通过 E-mail、MMS 或直接上传到 YouTube 等方式来分享视频，使用隐私控制来决定是分享给朋友还是每个人。上传视频的同时，可以继续使用手机，甚至可以继续录制和上传新的视频。

（5）图片上传。在线分享图片需要的操作步骤更少。完成照相后，当浏览图片或选择 Google 在线图片服务 Picasa 时，只需轻点“分享”就会拥有 1GB 的免费图片存储空间。

（6）更快、更兼容的浏览器。Android 的基于 Webkit 内核的浏览器带来了重要的调速装置（SpeedPumb），这得益于新的 Webkit 渲染引擎和优化的 Java 脚本编译器（SquireIFish）。当使用包含大量 Java 脚本的复杂 Web 应用时，可以体验到更佳的性能。除提高速度外，Android 的浏览器还支持 Web 页面内的复制和粘贴操作，用户可以选中文本并复制，然后粘贴到搜索框中进行搜索。

（7）Voice Search 语音搜索。带有语音识别技术的 Google 手机已于 2008 年 11 月面世，它支持语音搜索功能。该功能增强了默认的搜索能力，已超过纯文本搜索。当你大声说出要搜索的内容后，Android 将上传数字信号并记录到 Google 服务器中。在服务器中，语音识别技术能将语音转化为特定的文本搜索，使之通过 Google 搜索引擎和地理位置的筛选，将结果反馈到手机设备。

（8）立体声蓝牙和免提电话。除了增强的免提电话体验，Android 还支持立体声蓝牙（A2DP 和 AVCRP）并有自动配对功能。

（9）强大的 GPS 技术。Android 内部提供了大量 GPS 组件，可以很轻松地获得设备当前的位置等信息，让导航等功能更加完美。



（10）Android 系统硬件检测。Android 可自动检测和修复 SD 卡的文件系统，允许第三方应用显示 Android 系统的硬件特征。为了让用户下载到与自己的设备更匹配的应用，我们可以检测用户设备的硬件信息，让满足应用要求的设备安装该程序，当更多的 Android 设备建立在不同的硬件上时，这个功能会显得很实用。

2. Android 的系统架构

我们对 Android 的特点以及它为什么会如此受欢迎已经有了初步的了解。下面将讨论 Android 的系统构架，我们先来看看 Android 的体系结构。

Android 分为 4 层，从高到低分别是应用层、应用框架层、系统运行库层和 Linux 内核层。下面分别进行介绍。

（1）应用层。它是用 Java 语言编写的运行在虚拟机上的程序。其实，Google 最开始时就在 Android 系统中捆绑了一些核心应用，如 E-mail 客户端、SMS 短消息程序、日历、地图、浏览器和联系人管理程序等。

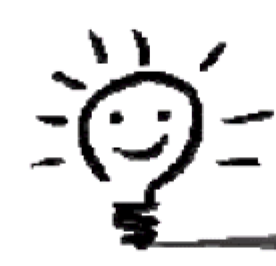
（2）应用框架层。这一层是编写 Google 发布的核心应用时所使用的 API 框架，开发人员同样可以使用这些框架来开发自己的应用，这样便简化了程序开发的架构设计，但是必须遵守其框架的开发原则。

在 Android 中提供了如下一些组件。

☑ 丰富而又可扩展的视图（View）：可以用来构建应用程序，它包括列表（List）、网格（Grid）、文本框（Text Box）、按钮（Button）以及可嵌入的 Web 浏览器。
☑ 内容提供器（Content Providers）：它可以让一个应用访问另一个应用的数据（如联系人数据库）或共享它们自己的数据。
☑ 资源管理器（Resource Manager）：提供非代码资源的访问，如本地字符串、图形和布局文件（Layout File）。
☑ 通知管理器（Notification Manager）：可以在状态栏中显示自定义的提示信息。
☑ 活动管理器（Activity Manager）：用来管理应用程序生命周期并提供常用的导航退回功能。
☑ 窗口管理器（Window Manager）：管理所有的窗口程序。
☑ 包管理器（Package Manager）：Android 系统内的程序管理。

（3）系统运行库层。当使用 Android 应用框架时，Android 系统会通过一些 C/C++库来支持使用的各个组件，并使其更好地为我们服务。

☑ Bionic 系统 C 库：C 语言标准库是系统最底层的库，C 库通过 Linux 系统来调用。
☑ 多媒体库（MediaFramework）：Android 系统多媒体库是基于 PacketVideo、OpenCORE，该库支持多种常见格式的音频、视频以及图片，如 MPEG4、MP3、AAC、AMR、JPG、PNG 等格式。
☑ SGL：2D 图形引擎库。
☑ SSL：位于 TCP/IP 协议与各种应用层协议之间，为数据通信提供支持。
☑ OpenGL ES 1.0：3D 效果的支持。
☑ SQLite：关系数据库。



☑ Webkit：Web 浏览器引擎。

☑ FreeType：位图（Bitmap）及矢量（Vector）。

每个 Java 程序都运行在 Dalvik 虚拟机之上。与 PC 机一样，每个 Android 应用程序都有自己的进程，Dalvik 虚拟机只执行.dex（Dalvik Executable）可执行文件。Java 程序通过编译后，还需要通过 SDK 中的 dx 工具转化成.dex 格式才能在虚拟机上正常执行。

Google 于 2007 年底正式发布了 Android ADK，作为 Android 系统的重要特性，Dalvik 虚拟机也第一次进入了人们的视野。它对内存的高效使用，以及在低速 CPU 上表现出的高性能，确实令人刮目相看。Android 系统可以简单地完成进程隔离和线程管理。每一个 Android 应用在底层都会对应一个独立的 Dalvik 虚拟机实例，其代码在虚拟机的解释下得以执行。

很多人认为 Dalvik 虚拟机是一个 Java 虚拟机，因为 Android 的编程语言恰恰就是 Java 语言。但是这种说法并不准确，因为 Dalvik 虚拟机并不是按照 Java 虚拟机的规范来实现的，两者并不兼容。它们有两个明显的不同：Java 虚拟机运行的是 Java 字节码，而 Dalvik 虚拟机运行的是.dex 文件。在 Java SE 程序中的 Java 类会被编译成一个或者获取相应的字节码文件（.class），然后打包到.jar 文件，而后 Java 虚拟机会从相应的.class 文件和.jar 文件中获取相应的字节码；Android 应用虽然也是使用 Java 语言编程，但是在编译成.class 文件后，还会通过一个工具（dx）将应用的所有.class 文件转换成一个.dex 文件，而后 Dalvik 虚拟机会从其中读取指令和数据。

Dalvik 虚拟机非常适合在移动终端上使用，相对于在桌面系统和服务器系统运行的虚拟机而言，它不需要很快的 CPU 计算速度和大量的内存空间。根据 Google 的测算，64MB 的内存已经能够让系统正常运转。其中 24MB 被用于底层系统的初始化和启动，另外 20MB 被用于启动高层服务。当然，随着系统服务的增多和应用功能的扩展，其所消耗的内存也越来越大。归纳起来，Dalvik 虚拟机有如下几个主要特征：

① 专有的.dex 文件格式。.dex 格式是 Dalvik 虚拟机专用的文件格式，为什么弃用已有的字节码文件（.class 文件）而采用新的格式呢？其原因如下：

☑ 每个应用中会定义很多类，编译完成后即会有很多相应的.class 文件，.class 文件中会有大量冗余信息，而.dex 文件格式会把所有的.class 文件内容整合到一个文件中。这样，除了减少整体的文件尺寸和 I/O 操作外，也提高了类的查找速度。

☑ 增加了对新的操作码的支持。

☑ 文件结构尽量简洁，使用等长的指令，借以提高解析速度。

☑ 尽量扩大只读结构的大小，借以提高跨进程的数据共享。

② dex 的优化。.dex 文件的结构是紧凑的，但是如果还想进一步提高运行性能，就需要对.dex 文件进一步优化。优化主要针对以下几个方面：

☑ 调整所有字段的字节序（LITTLE_ENDIAN）和对齐结构中的每一个域。

☑ 验证.dex 文件中的所有类。

☑ 对一些特定的类和方法里的操作码进行优化。

③ 基于寄存器。相对于基于堆栈实现的虚拟机，基于寄存器实现的虚拟机虽然在硬件、通用性上要差一些，但是它在代码的执行效率上却更胜一筹。

④ 一个应用，一个虚拟机实例，一个进程。每一个 Android 应用都运行在一个 Dalvik 虚拟机实例中，而每一个虚拟机实例都是一个独立的进程空间。虚拟机的线程机制、内存分配和管理、Mutex 等的实现都依赖底层操作系统。所有 Android 应用的线程都对应一个 Linux 线程，虚拟机因而可以更多地依赖操作系统的线程调度和管理机制。不同的应用在不同的进程空间里运行，对不同来源的应用都使用不同的 Linux 用户来运行，可以最大程度地保护应用的安全和独立运行。

（4）Linux 内核层。Android 的核心系统服务基于 Linux 2.6 内核，如安全性、内存管理、进程管理、网络协议栈和驱动模型等都依赖于该内核。Linux 内核同时也作为硬件和软件栈之间的抽象层。

Android 更多的是需要一些与移动设备相关的驱动程序，主要的驱动如下。

- ☑ 显示驱动（Display Driver）：基于 Linux 的帧缓冲（Frame Buffer）驱动。
- ☑ 键盘驱动（KeyBoard Driver）：作为输入设备的键盘驱动。
- ☑ Flash 内存驱动（Flash Memory Driver）：基于 MTD 的 Flash 驱动。
- ☑ 照相机驱动（Camera Driver）：基于 Linux 的 V412（Video for Linux）驱动。
- ☑ 音频驱动（Audio Driver）：基于 ALSA（Advanced Linux Sound Architecture）的高级 Linux 声音体系驱动。
- ☑ 蓝牙驱动（Bluetooth Driver）：基于 IEEE 802.15.1 标准的无线传输技术。
- ☑ WiFi 驱动：基于 IEEE 802.11 标准的驱动。
- ☑ Binder IPC 驱动：Android 的一个特殊的驱动程序，具有单独的设备节点，提供进程间通信的功能。
- ☑ 电源管理（Power Management）：如电池电量等。

3. Android 应用程序框架

前文已经对 Android 的系统构架进行了详细剖析，Android 分为应用层、应用框架层、系统运行库层和 Linux 内核层。我们在开发应用时都是通过应用程序框架与 Android 底层进行交互的，接触最多的就是应用框架层了。

什么是应用程序框架呢？框架可以说是一个应用程序的核心，是所有参与开发的程序员共同使用和遵守的约定，大家在其约定上进行必要的扩展，但程序始终保持主体结构的一致性。其作用是让程序始终清晰和一目了然，在满足不同需求的同时又不互相影响。

Android 系统提供给应用开发者的本身就是一个框架，所有的应用开发都必须遵守这个框架的原则，我们在开发应用时就是在这个框架上进行扩展。下面来看看 Android 这个框架的具体功能。

- ☑ android.app：提供高层的程序模型和基本的运行环境。
- ☑ android.content：包含对各种设备上的数据进行访问和发布。
- ☑ android.database：通过内容提供者浏览和操作数据库。
- ☑ android.graphics：底层的图形库，包括画布、颜色过滤、点、矩阵，可以将它们直接绘制到屏幕上。
- ☑ android.location：定位相关服务的类。



- ☑ android.media：提供一些类管理多种音频、视频的媒体接口。
- ☑ android.net：提供帮助网络访问的类，超过通常的 java.net.*接口。
- ☑ android.os：提供系统服务、消息传输和 IPC 机制。
- ☑ android.opengl：提供 OpenGL 的工具。
- ☑ android.provider：提供访问 Android 内容提供者的类。
- ☑ android.telephony：提供与拨打电话相关的 API 交互。
- ☑ android.view：提供基础的用户界面接口框架
- ☑ android.util：涉及工具性的方法，如时间、日期的操作等。
- ☑ android.webkit：默认浏览器操作接口。
- ☑ android.widget：包含各种 UI 元素（大部分是可见的）在应用程序的布局中使用。

任务 2　OMS 基本介绍

【任务情境】

OMS（Open Mobile System，面向移动互联网的开放型移动智能终端软件平台），包括基于 Linux 2.6 内核的移动终端下层操作系统、上层应用软件、中间件、Java 虚拟机、硬件参考设计以及基于 WebKit 的各类应用。它不光具有强大的兼容性、扩展性和安全性，以及简单易用、友好的人机界面，而且拥有完全自主的知识产权。此外，OMS 还拥有开放统一的 API 开发接口、完备的集成开发环境和活跃的在线生态环境，极大地方便了移动应用的开发。

OMS 的可移植性将使该软件平台在其他领域具有广泛的应用，如航空航天、军事、制造业等。

【相关知识】

1. OPhone 介绍

OPhone 是基于 Linux 的面向移动互联网的终端基础软件及系统解决方案。由于 OPhone 与 Android 兼容，都是基于 Java 开发的，因此可以同时使用 OMS API 和 Android API 来开发 OMS 应用。任何用 Android API 开发的应用都可以在 OMS 终端上正确运行。然而，由扩展的 OMS API 开发的应用不能在 Android 终端上运行，因为这些 OMS API 是 OMS 平台独有的，而且在运行时是必需的。

OPhone 是指采用了 OMS 智能操作系统的手机。为了突破 TD 终端瓶颈，以及促进手机终端与中国移动的网络和应用服务进行无缝对接，中国移动在 Android 操作系统基础上自主开发了 OMS 系统，该系统直接内置了中国移动的服务菜单、音乐随身听、手机导航、号簿管家、139 邮箱、飞信、快讯和移动梦网等特色业务。



2. Widget 介绍

OMS 除了支持基于 Java 的应用外，还支持 Widget 应用开发。Widget 应用是 OMS 的精华，而 Android 从 1.5 版本开始同样支持 Widget 应用开发，但是所采用的标准和 OMS 不同。

Widget 应用采用了 JIL（Joint Innovation Lab）Widget 标准。JIL Widget 是一款采用 HTML、JavaScript 和 CSS 等网络技术的应用程序。Widget 应用是在 Widget 引擎上运行的独立的应用程序。Widget 已经成为手机上非常流行的技术，可以为用户带来良好的移动互联网体验，可随时随地获取有用的资讯，如天气预报、股票信息、头条新闻等。从用户的角度来看，Widget 应用和 OPhone 应用没什么区别；实际上二者大不相同。OPhone 应用采用 Java 技术，而 Widget 应用则采用 HTML、JavaScript 和 CSS 等网络技术。相比较而言，Widget 应用的开发更加方便快捷。此外，JIL Widget 还提供了许多 JavaScript API 扩展的 Widget 应用能力，如访问手机电话本、手机文件系统等。

【项目小结】

本项目主要介绍了与 Android 相关的一些基本概念，同时分析了 Android 系统的特点及其功能，简单介绍了中国移动的 OMS 操作系统。在介绍 Android 基本概念时重点介绍了 Android 系统架构和应用框架，其中应用层是使用 Java 语言编写的一些运行在虚拟机上的程序，应用框架层是开发应用时接触最为紧密的一层，在开发应用程序时必须遵守其规则，才能保证所开发的应用程序能在 Android 上安全地运行。大家应着重理解这两层，这样才能开发出效率更高的应用程序。

现如今以 Android 平台为系统的 Linux 手机越来越多，Android 将在移动开发中具有更广阔的前景。然而要想成为一名优秀的 Android 手机开发者，还需要从基础做起，希望大家好好掌握本项目中讲叙的内容。

综合实训一　Android 发展大事记

2005 年事件

☑ Google 收购了成立仅 22 个月的高科技企业 Android。

2007 年事件

☑ 11 月 5 日，Google 公司正式向外界展示 Android 操作系统。

☑ 11 月 5 日，Google 与 34 家手机制造商、软件开发商、电信运营商和芯片制造商共同创建“开放手持设备联盟”。

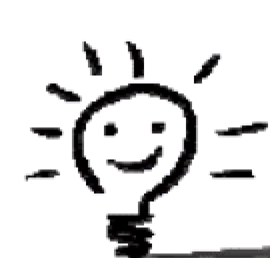
2008 年事件

☑ 5 月 28 日，Patrick Brady 于 Google I/O 大会上提出 Android HAL 架构图。

☑ 8 月 18 日，Android 获得美国联邦通信委员会的批准。

☑ 9 月 22 日，Google 正式对外发布第一款 Android 手机——T-Mobile G1。

☑ 9 月 23 日，Google 正式对外发布 Android1.0 操作系统。



- ☑ 9 月 24 日，全球业界都表示不看好 Android 操作系统，并且声称最多 1 年，Android 就会被 Google 关闭。

2009 年事件

- ☑ 4 月 30 日，Android1.5 正式发布。
- ☑ 5 月 10 日，HTC G1 和 HTC G2 市场大卖，成为仅次于 iPhone 的热门机型。
- ☑ 9 月 25 日，Android1.6 正式发布。Android1.6 将增加对 CDMA 网络的支持，重新设计了 Android Market，以及进一步加强了操作系统的搜索功能。
- ☑ 9 月 29 日，HTC Hero G3 广受欢迎，成为全球最受欢迎的机型。
- ☑ 10 月 28 日，Android2.0 智能手机操作系统正式发布。
- ☑ 11 月 10 日，由于 Android 的火热，Android 平台出现第一个恶意间谍软件——Mobile Spy，该程序会自动记录用户所输入的任何信息并发送到黑客的邮箱中，还可以通过摄像头录下用户的所有操作过程。

2010 年事件

- ☑ 1 月 7 日，Google 发布了旗下第一款自主品牌手机——Nexus one（G5）。
- ☑ 1 月，Google 与 Linux 基金会就 Linux 内核的同步和维护意见不一致而产生了矛盾。
- ☑ 2 月 3 日，Linux 内核开发者 Greg Kroah-Hartman 将 Android 的驱动程序从 Linux 内核“状态树”上除去，从此，Android 与 Linux 开发分道扬镳。
- ☑ 5 月 19 日，Google 正式对外发布 Android2.2 智能操作系统。
- ☑ 5 月 20 日，Google 对外正式展示了搭载 Android 系统的智能电视—Google TV，该电视为全球首台智能电视。Android 手机可以当做 Google TV 的遥控器。
- ☑ 7 月 1 日，Google 宣布正式与雅虎、亚马逊合作，并且在 Android 上推出多项 Kindle 服务和雅虎服务。
- ☑ 7 月 9 日，据美国 NDP 集团调查显示，Android 系统已占据了美国手机市场约 28%的份额，全球约 17%的市场份额。
- ☑ 8 月 12 日，Android 平台出现第一个木马病毒：Trojan-SMS.AndroidOS.FakePlayer.a，该木马病毒会伪装成应用程序，当用户不小心安装之后，它便会疯狂地发送短信，使用户的手机开通高额的收费服务。
- ☑ 9 月，Android 应用数量超过 9 万个。
- ☑ 9 月 21 日，Google 对外公布数据，每日 Android 设备的新用户数达到 20 万。
- ☑ 10 月 26 日，Google 宣布 Android 达到第一个里程碑：电子市场上的 Android 应用数量达到 10 万个。
- ☑ 12 月 7 日，Google 正式发布 Android 2.3 操作系统。

2011 年事件

- ☑ 1 月，Google 对外宣布 Android Market 上的应用数量超过 20 万。
- ☑ 1 月，Google 对外公布数据，每日 Android 设备的新用户数达到 30 万。
- ☑ 2 月 2 日，Android 3.0 正式发布。
- ☑ 2 月 3 日，Google 发布了专用于平板电脑的 Android 3.0 Honeycomb 系统，它带来

了很多激动人心的新特性。这是首个基于 Android 的平板电脑专用操作系统。

☑ 6 月，Android 在日本的智能手机操作系统市场占有率达到 57%。

☑ 7 月，Android 在欧洲的智能手机操作系统市场占有率达到 22.3%。

☑ 7 月，Google 对外公布数据，Android 每天的新用户达到 55 万，Android 设备用户总数达到 1.35 亿。

☑ 8 月，Google 收购摩托罗拉移动。

☑ 8 月，Google 宣布 Android Market 上的应用数量超过 30 万。

☑ 8 月 2 日，Android 手机已占据全球智能机市场 48%的份额，并在亚太地区市场占据统治地位，终结了 Symbian（塞班）系统的霸主地位，跃居全球第一。

☑ 10 月 19 日，Google 正式发布 Android 4.0 操作系统。

☑ 11 月，Android Market 上提交审核的应用程序数量达到 50 万。随后，Google 对 Android Market 上的应用程序进行了大清理，据统计，此次共清理了约 18 万个应用程序，包括流氓应用、病毒软件、侵犯版权、低质量、滥竽充数的各种程序，Google 将这一系列应用删除后，使得 Android 市场中优质应用程序总数达到 31.5 万。

☑ 11 月 15 日，Android 在中国大陆的智能手机操作系统市场占有率达到 58%。

☑ 11 月 18 日，据美国 NPD 调查数据显示，Android 和 iOS 平台上的游戏占有率首度超过任天堂的 DS 掌机和索尼的 PSP 掌机，手机游戏玩家也超过了掌机玩家，游戏开发商更倾向于在 Android 和 iOS 手机上开发游戏。

☑ 11 月 18 日，Google 报告显示，通过 Google 服务器激活的 Android 设备用户总数已经超过 2 亿，每天通过 Google 服务器激活的新用户数超过 55 万，而这仅仅是通过 Google 服务器激活的用户设备数。

☑ 11 月 20 日，Google 宣布启动了 Android Market 应用审核、取缔、清扫行为，定期对电子市场上存在的不合格、低质量、违法恶意的应用程序进行清理，保证 Android 优质应用程序的增长。

☑ 12 月 9 日，Google 对外宣布，Android 达到另一个里程碑，Android 电子市场即 Android Market 的累积下载量已经突破 100 亿次，平均每月的下载量为 10 亿次。

☑ 12 月 18 日，Google 移动事业部副总裁通过 Andy Rubin 表示，每天激活的 Android 设备已达到 70 万部。

☑ 12 月 26 日，Andy Rubin 通过 Twitter 宣布，圣诞节的前两天 24 日和 25 日，共有 370 万部 Android 设备被激活。

【Android 系统 VS 苹果系统】

Android 系统的优点：

（1）能看 RM 等多种格式的视频，而苹果系统则不能。

（2）能看 TXT 等多种格式的数据，而苹果系统则不能。

（3）能上网看视频，而苹果系统则不能。

（4）能有大量免费的第三方软件、游戏使用，而苹果系统则不能。

（5）能很方便地和 PC 连接随意传输文件，而苹果系统则不能。

（6）强大、开放的硬件规格有较大拓展的空间，而苹果系统则不能。



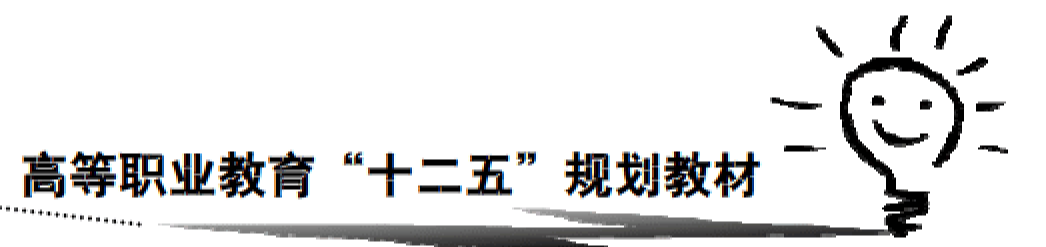
苹果系统的优点：

（1）安卓系统目前没有针对性，苹果系统是原创的系统。

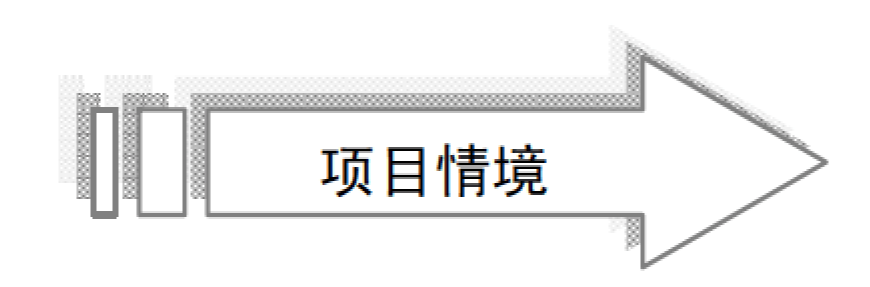
（2）安卓软件是一款智能手机使用的软件，苹果系统在自带的软件和游戏中表现很出色。

（3）安卓屏幕在分辨率和质量上很难和苹果比肩，苹果提供了更精准的颜色，以及更大的可视角度，用户体验也更为灵敏。

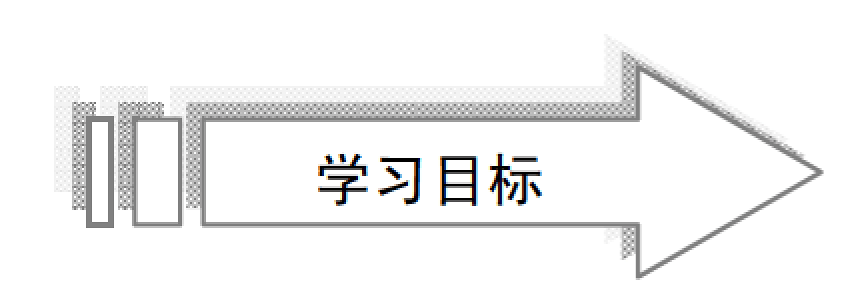
（4）安卓系统做工一般、外观平常，苹果系统的外观和做工比较精美。



项目二
Android 初体验



本项目将介绍 Android 系统应用程序框架及开发环境的搭建，让读者对 Android 平台有一个初步了解，之后将开发第一个 Android 程序 HelloAndroid，并通过对该程序的简单分析，带领读者步入 Android 开发的大门。



- 能够搭建 Android 系统应用程序框架及开发环境
- 掌握在 Android 环境下搭建 HelloAndroid 项目，并熟悉相关项目的搭建流程
- 了解 HelloAndroid 项目的程序框架和代码内容
- 了解 DDMS 监控 Android 应用程序的方法，熟悉如何调试 Android 程序

任务 3　Android 开发环境的搭建

【任务情境】

本任务主要讲述如何构建基于 Eclipse 的 Android 开发环境，并对开发环境进行测试。

【相关知识】

1. JDK 的下载与安装

（1）登录 SUN 官方网站 http://java.sun.com/javase/downloads，如图 2-1 所示，单击 Download 按钮进入下载界面，下载最新的 JDK 安装程序。

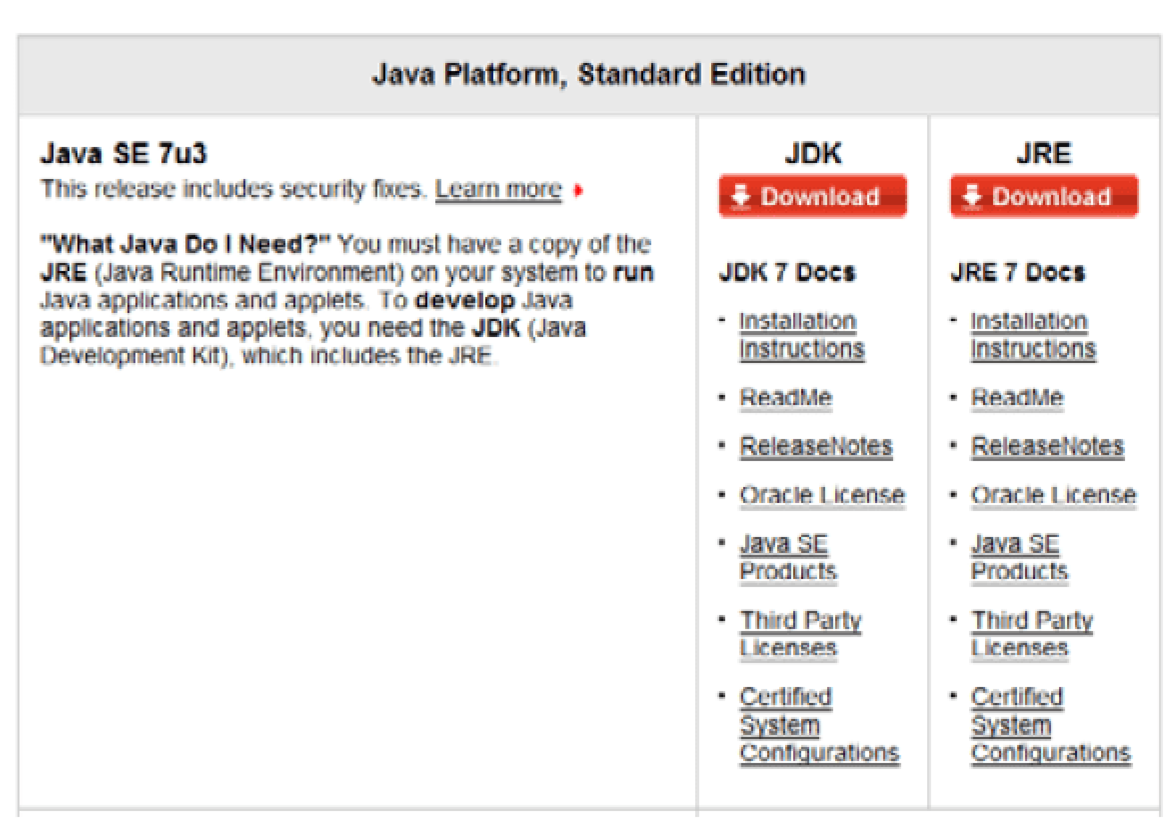
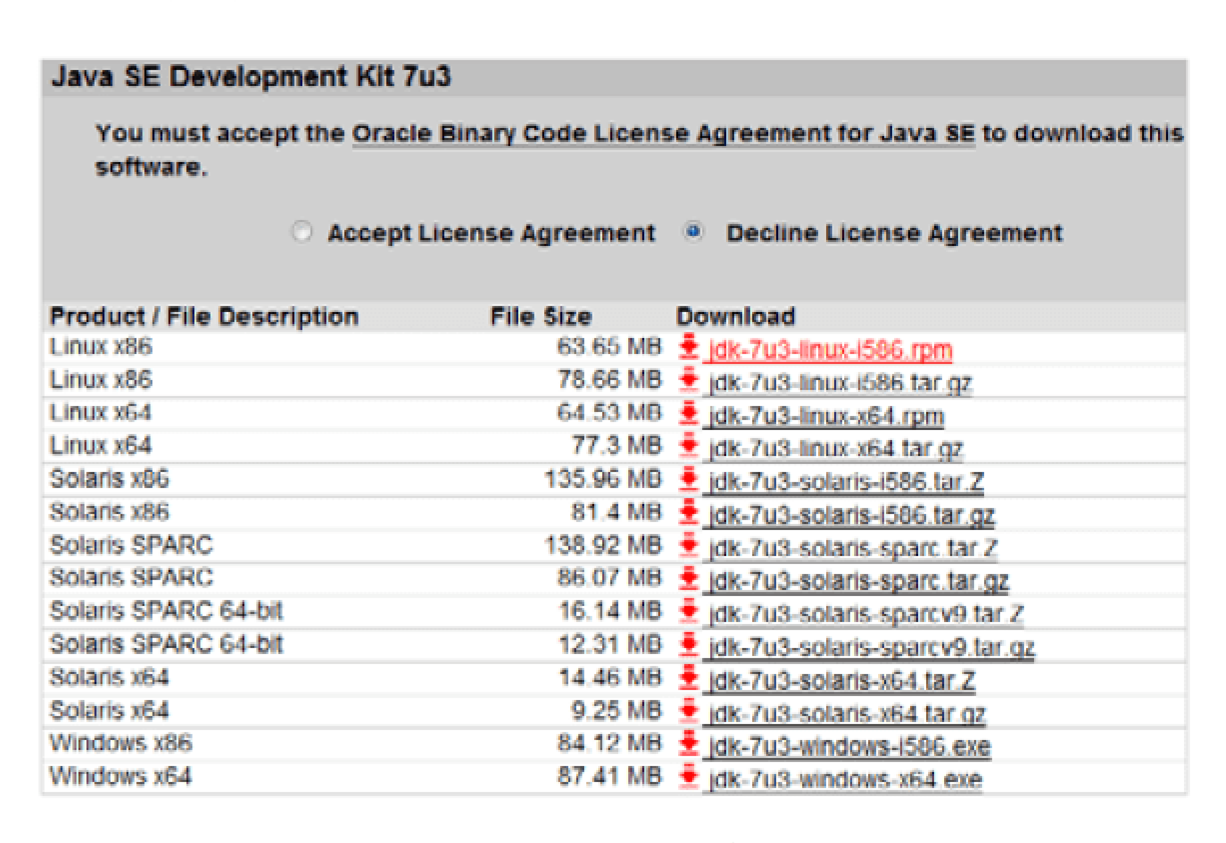


图 2-1　SUN 官网主页

（2）进入下载页面后，首先选择 Accept License Agreement 选项，然后根据自己的电脑和操作系统在列表框中选择对应的 JDK 文件进行下载，32 位操作系统选择 x86，64 位操作系统选择 x64，如图 2-2 所示。



Java SE Development Kit 7u3

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement　Decline License Agreement

Product / File Description	File Size	Download
Linux x86	63.65 MB	jdk-7u3-linux-i586.rpm
Linux x86	78.66 MB	jdk-7u3-linux-i586.tar.gz
Linux x64	64.53 MB	jdk-7u3-linux-x64.rpm
Linux x64	77.3 MB	jdk-7u3-linux-x64.tar.gz
Solaris x86	135.96 MB	jdk-7u3-solaris-i586.tar.Z
Solaris x86	81.4 MB	jdk-7u3-solaris-i586.tar.gz
Solaris SPARC	138.92 MB	jdk-7u3-solaris-sparc.tar.Z
Solaris SPARC	86.07 MB	jdk-7u3-solaris-sparc.tar.gz
Solaris SPARC 64-bit	16.14 MB	jdk-7u3-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	12.31 MB	jdk-7u3-solaris-sparcv9.tar.gz
Solaris x64	14.46 MB	jdk-7u3-solaris-x64.tar.Z
Solaris x64	9.25 MB	jdk-7u3-solaris-x64.tar.gz
Windows x86	84.12 MB	jdk-7u3-windows-i586.exe
Windows x64	87.41 MB	jdk-7u3-windows-x64.exe

图 2-2　JDK 下载页面



（3）下载解压后，双击 JDK 安装程序 jdk-7u3-windows-x64.exe，根据提示将 JDK 安装到默认目录下，这里将其安装到 C:\Program Files\Java\jdk1.7.0_03\。按照文件指引一步步安装即可，如图 2-3 所示。



图 2-3　JDK 安装界面

（4）检查 JDK 是否安装成功。打开“cmd”命令窗口，输入 java -version 查看 JDK 的版本信息，如图 2-4 所示。



图 2-4　查看 JDK 版本信息

2. Eclipse 的下载与安装

（1）登录 Eclipse 下载页面 http://www.eclipse.org/downloads/，下载与计算机相对应的版本。常用的还有 Eclipse IDE for Java EE Developers 和 Eclipse IDE for Java Developers 两个版本，其中第一个是企业版，包含的内容更多一些，这里下载另外一个，如图 2-5 所示。

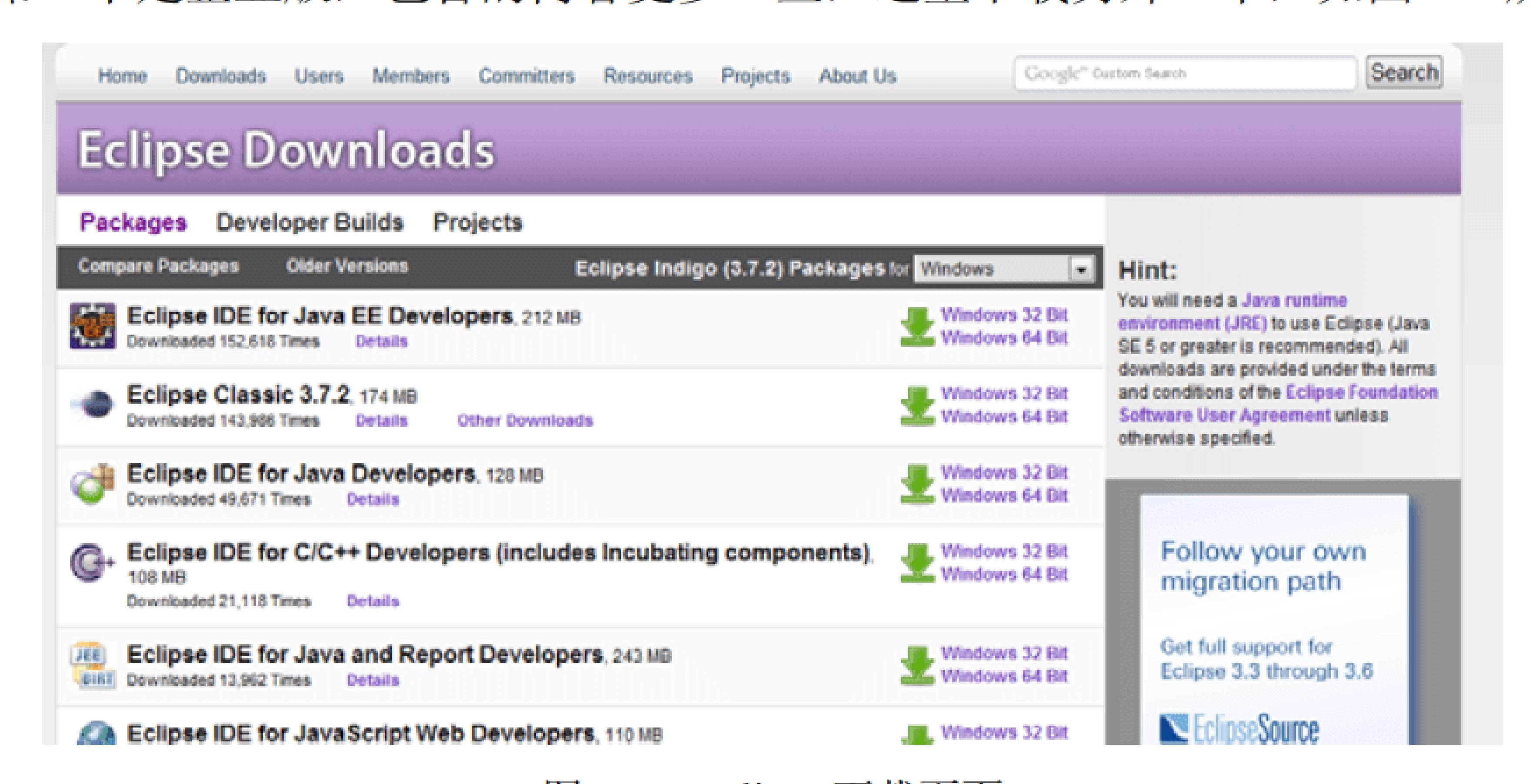


图 2-5　Eclipse 下载页面

（2）直接解压缩 eclipse-java-indigo-SR2-win32-x86-64.zip 到指定目录，如 D:\SOFTWARE\Android，如图 2-6 所示。

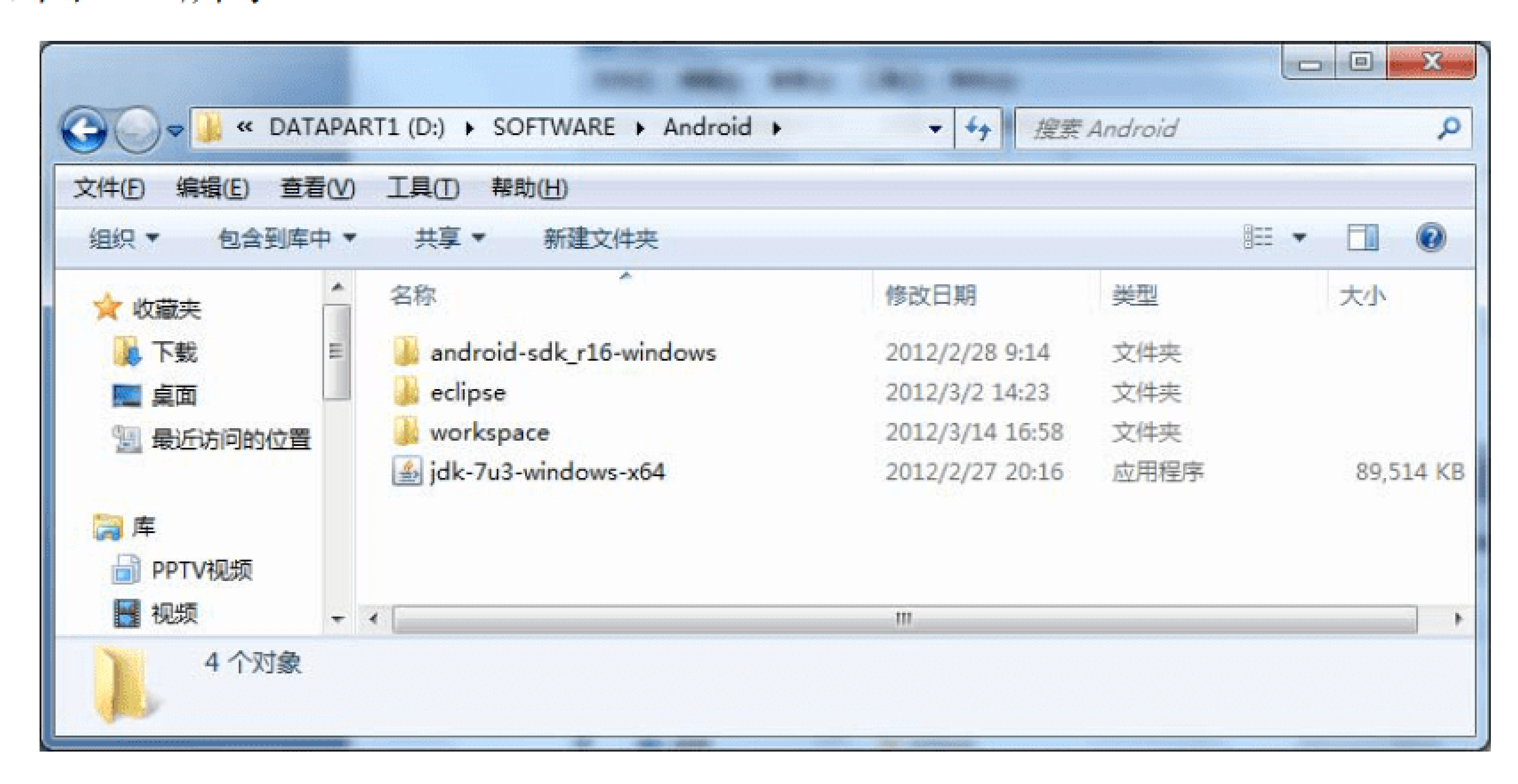


图 2-6　Eclipse 下载文件解压缩界面

（3）运行 eclipse.exe，设置 Workspace，即指定一个开发目录，如图 2-7 所示。

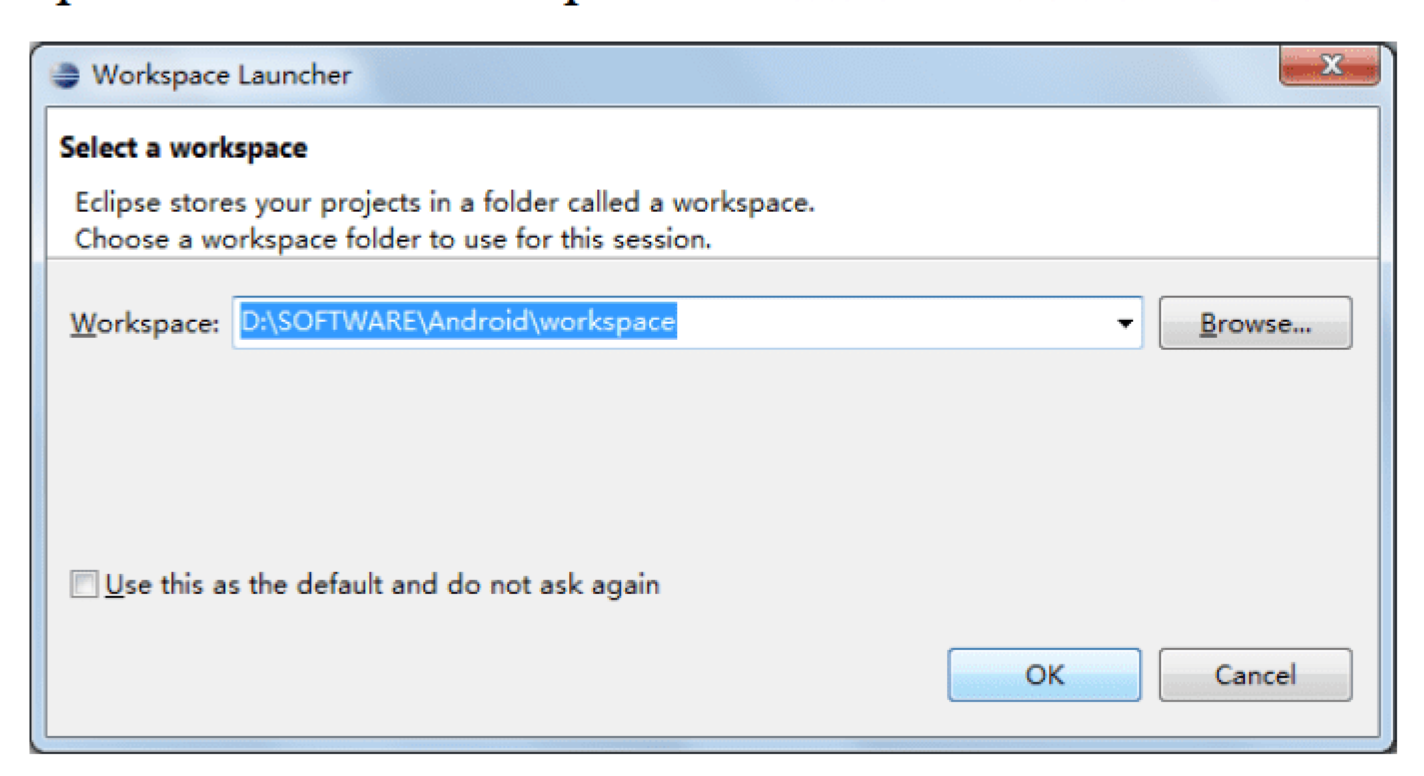


图 2-7　Eclipse 文件安装界面

（4）出现如图 2-8 所示的页面，Eclipse 安装完毕。

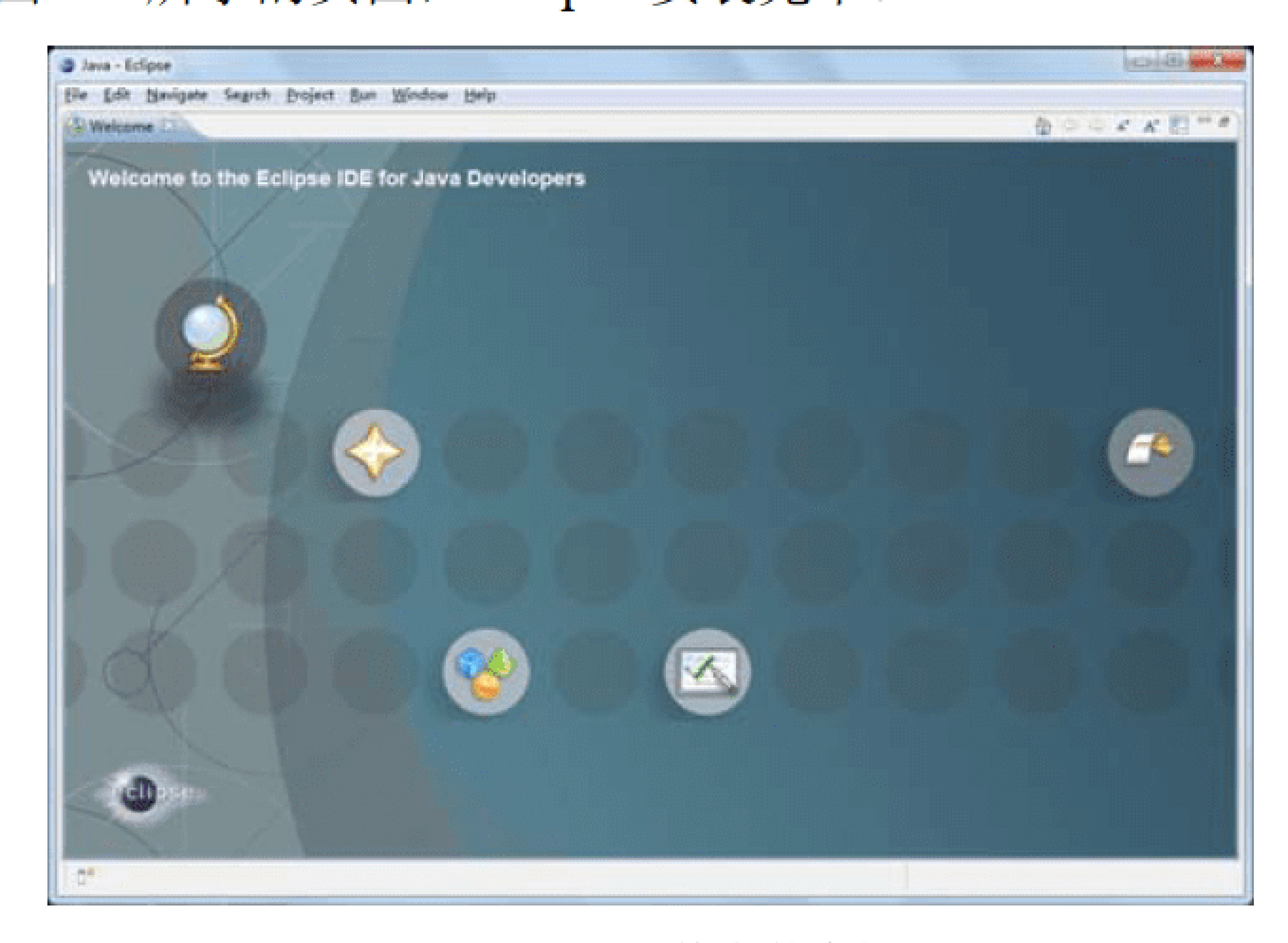


图 2-8　Eclipse 文件安装完毕界面



3. SDK 的下载与安装

（1）登录 Android 开发者网站 http://developer.android.com/index.html，单击 Developer 下拉菜单，选择 Get the SDK，如图 2-9 所示，单击 Download the SDK for Windows 按钮即可。

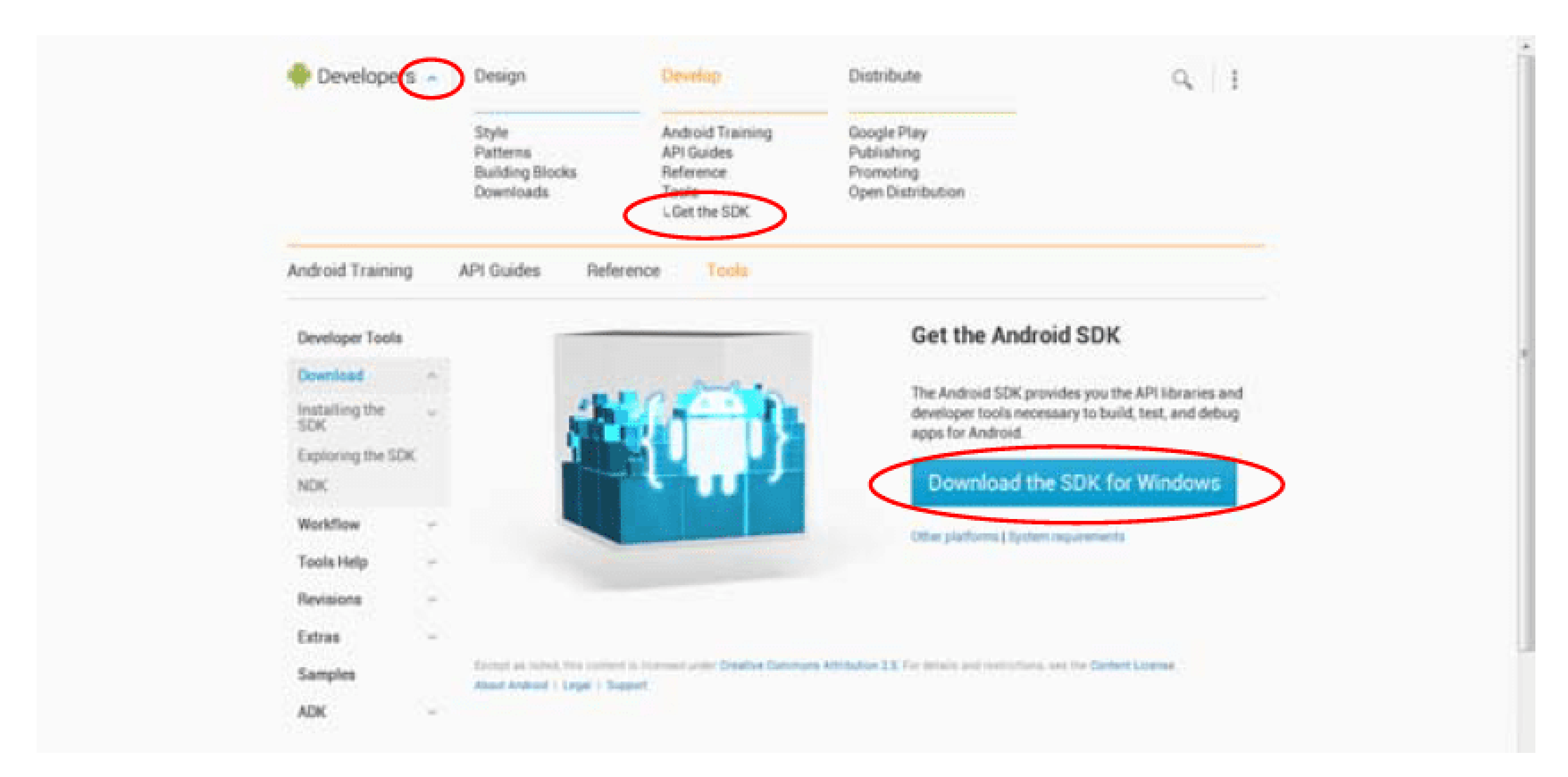


图 2-9　SDK 的下载页面

（2）Android SDK 同 Eclipse 一样，直接解压缩就可以，这里将其解压缩到 D:\SOFTWARE\Android 中，如图 2-10 所示。

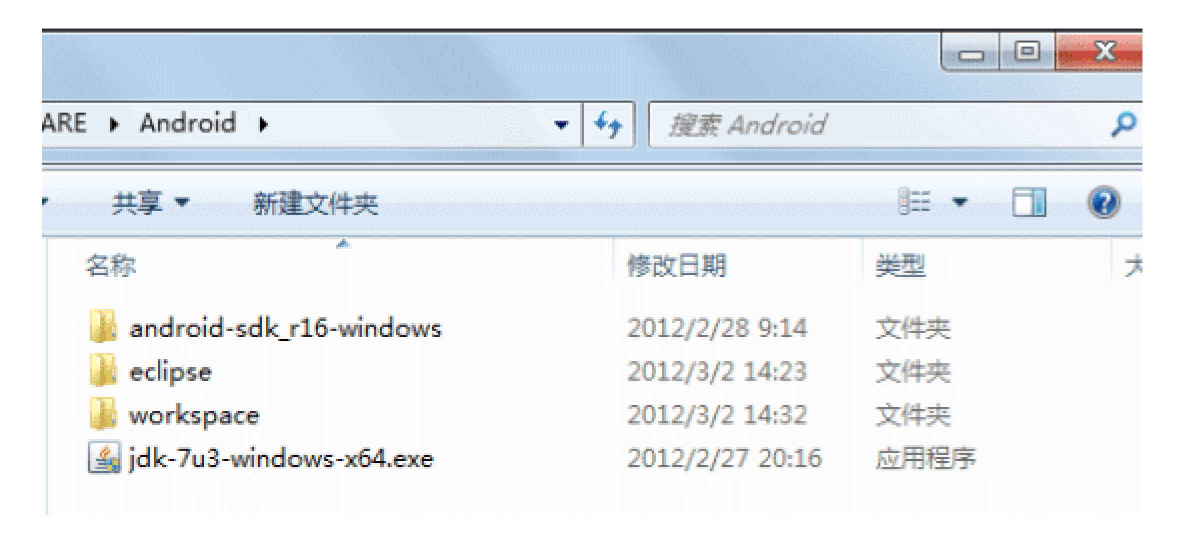


图 2-10　SDK 下载文件的解压缩界面

（3）配置：将 Android SDK 中的 tools 绝对路径添加到系统 PATH 中。打开“计算机”→“系统”→“高级系统设置”，选择“环境变量”，将 SDK 中 tools 的绝对路径添加到环境变量值 PATH 中，其值的具体编辑如下：C:\Program Files (x86)\IDM Computer Solutions\Ultra Edit\;D:\SOFTWARE\Android\android-sdk_r16-windows\android-sdk-windows\tools，如图 2-11 所示。

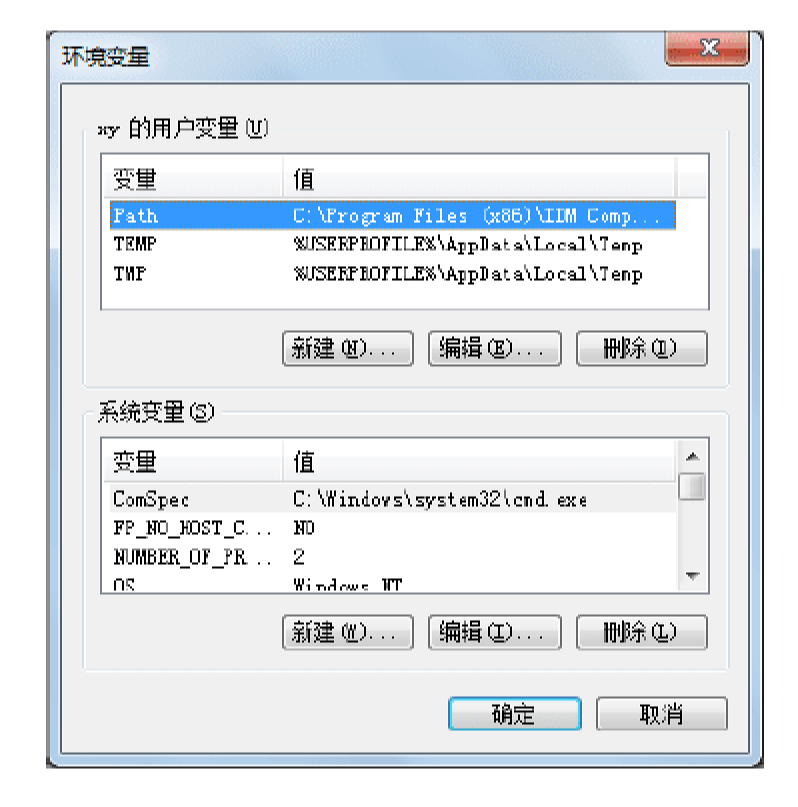
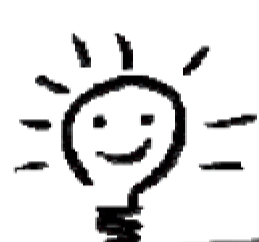


图 2-11　SDK 的配置运行设置界面



（4）单击“确定”按钮后，重新启动计算机，然后进入“cmd”命令窗口，检查 SDK 是不是安装成功。运行 android -h，如果有如图 2-12 所示的输出，即表明安装成功。

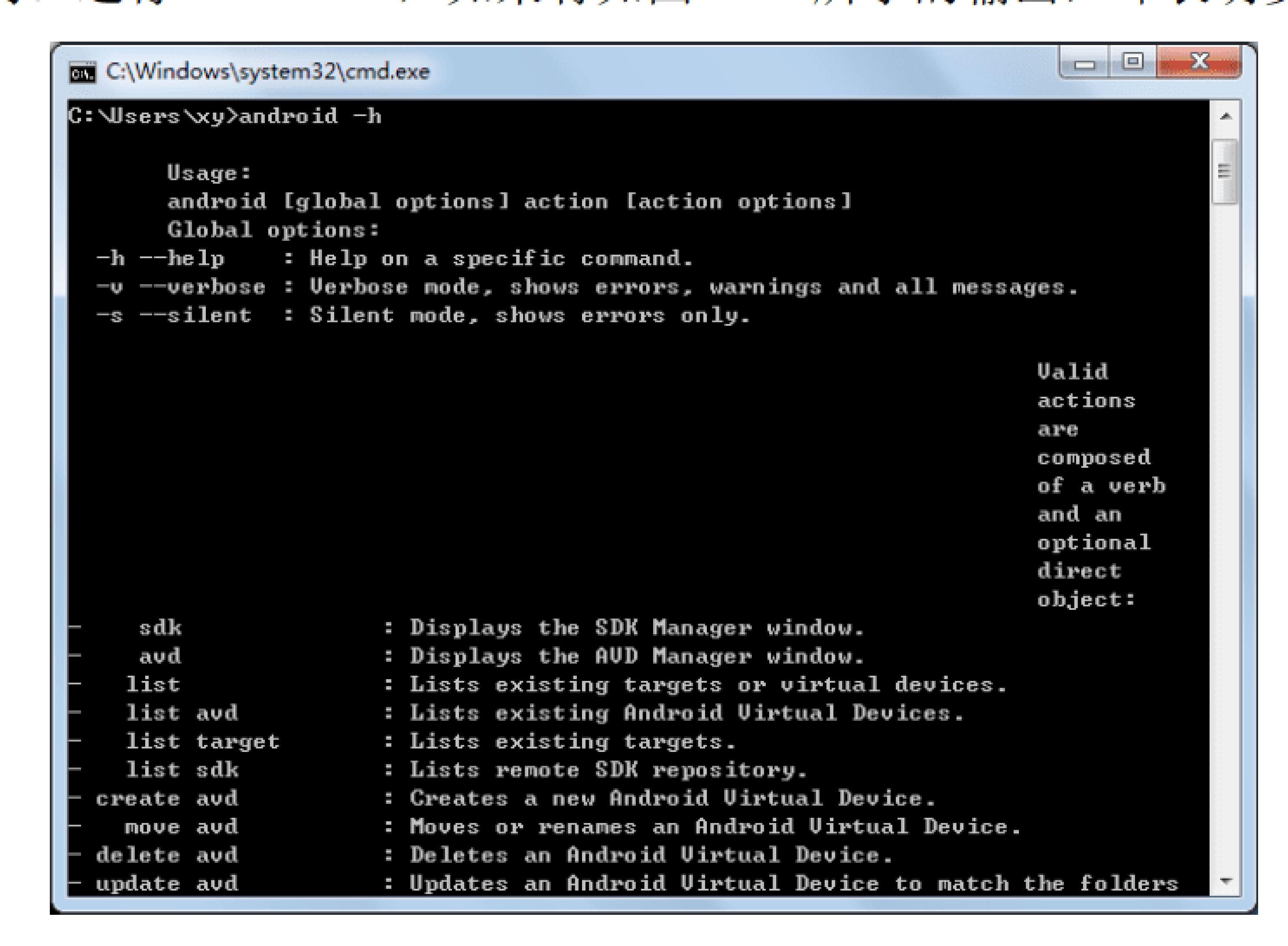


图 2-12　SDK 文件安装成功界面

4. ADT 的下载与安装

（1）登录 Android 开发者网站 http://developer.android.com/sdk/eclipse-adt.html，打开 SDK 下载页面，选择 ADT 下载部分下载最新版本。

（2）将下载的 ADT 解压到 D:\SOFEWARE\Android，如图 2-13 所示。

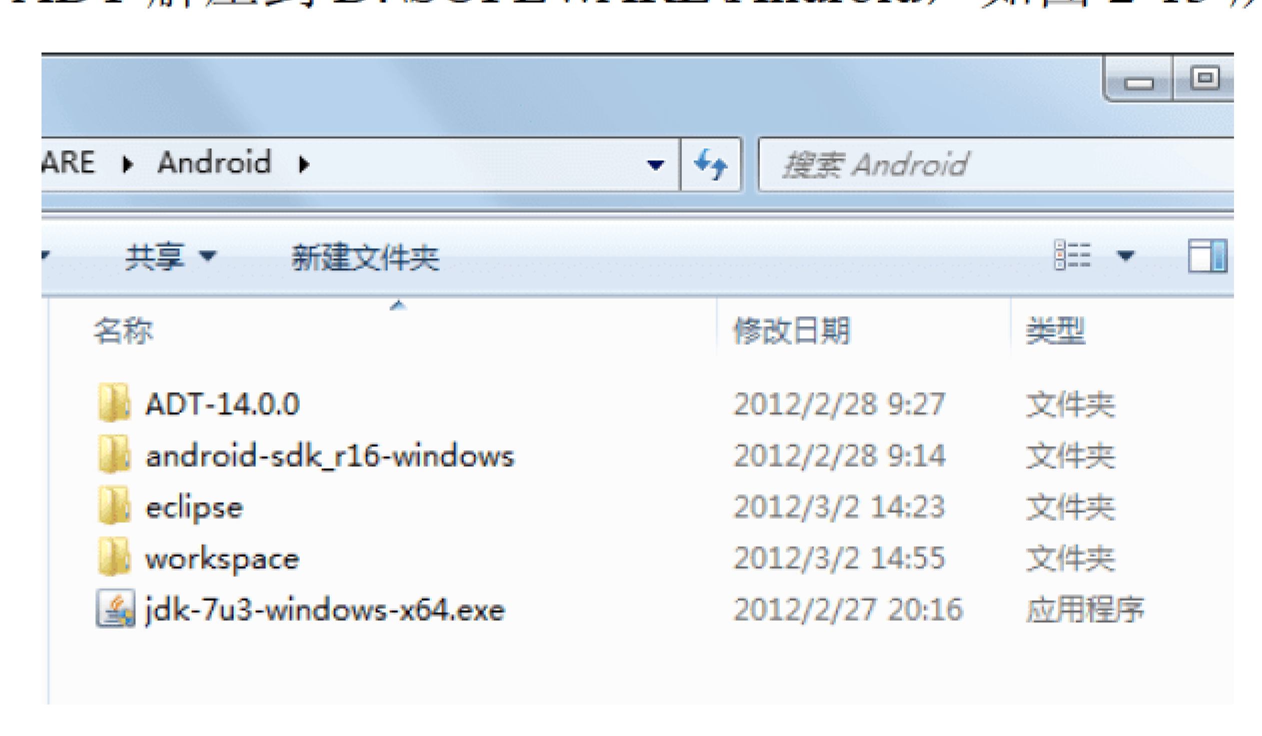


图 2-13　ADT 下载文件解压缩界面

（3）接下来，安装 ADT。打开 Eclipse，选择菜单 Help→Install New Software，进入 Available Software 选项卡，单击 Add 按钮，在 Work with 中输入 http://dl-ssl.google.com/android/eclipse，此时，在 Name 框中会出现 Developer Tools，单击展开后选中 Android DDMS 和 Android Development Tools 复选框，如图 2-14 所示，单击 Next 按钮，然后根据提示进行安装即可。

（4）重启 Eclipse，设定 SDK Location，选择菜单 Windows→Preferences，打开 Preferences 窗口，选择 Android，设定 SDK Location 为 SDK 的安装目录，单击 OK 按钮即可，如图 2-15 所示。



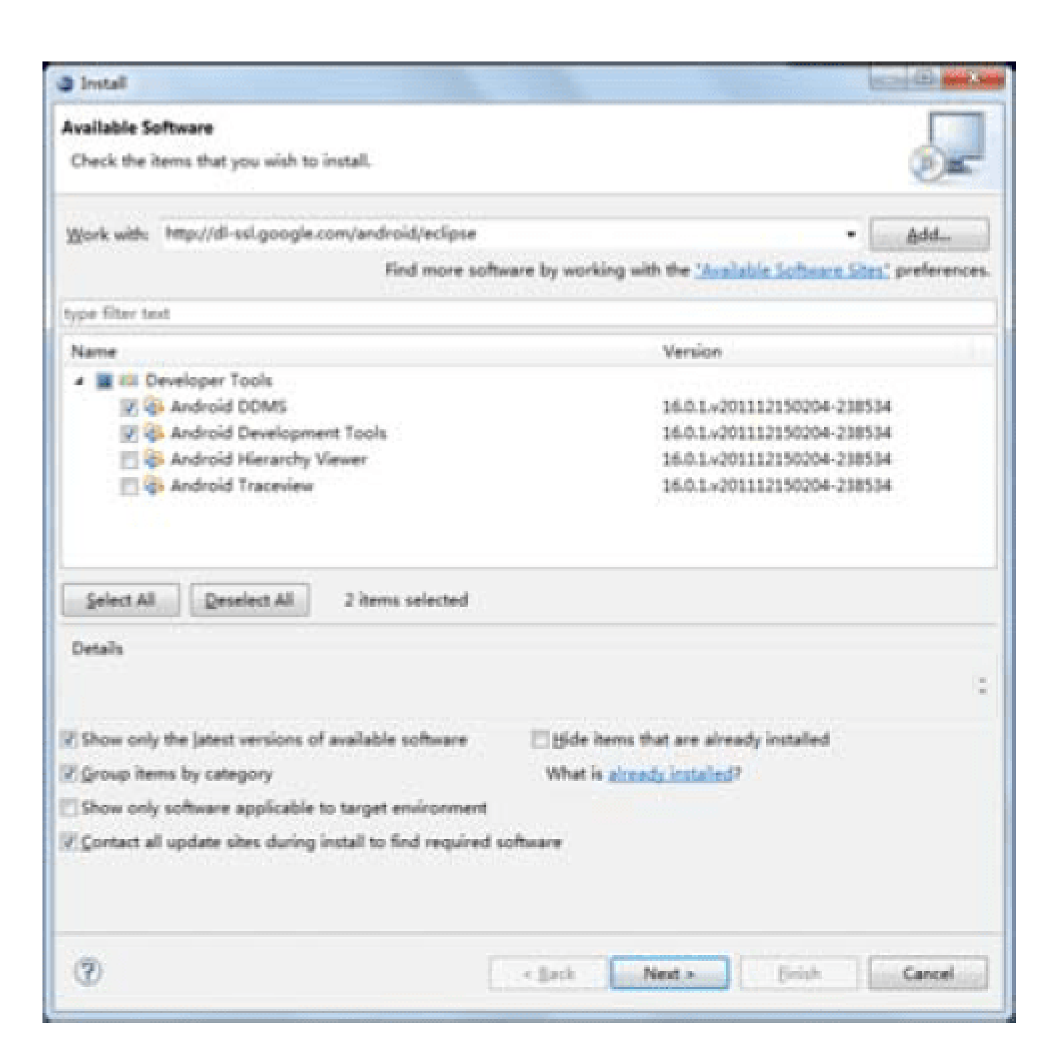


图 2-14 ADT 文件安装界面

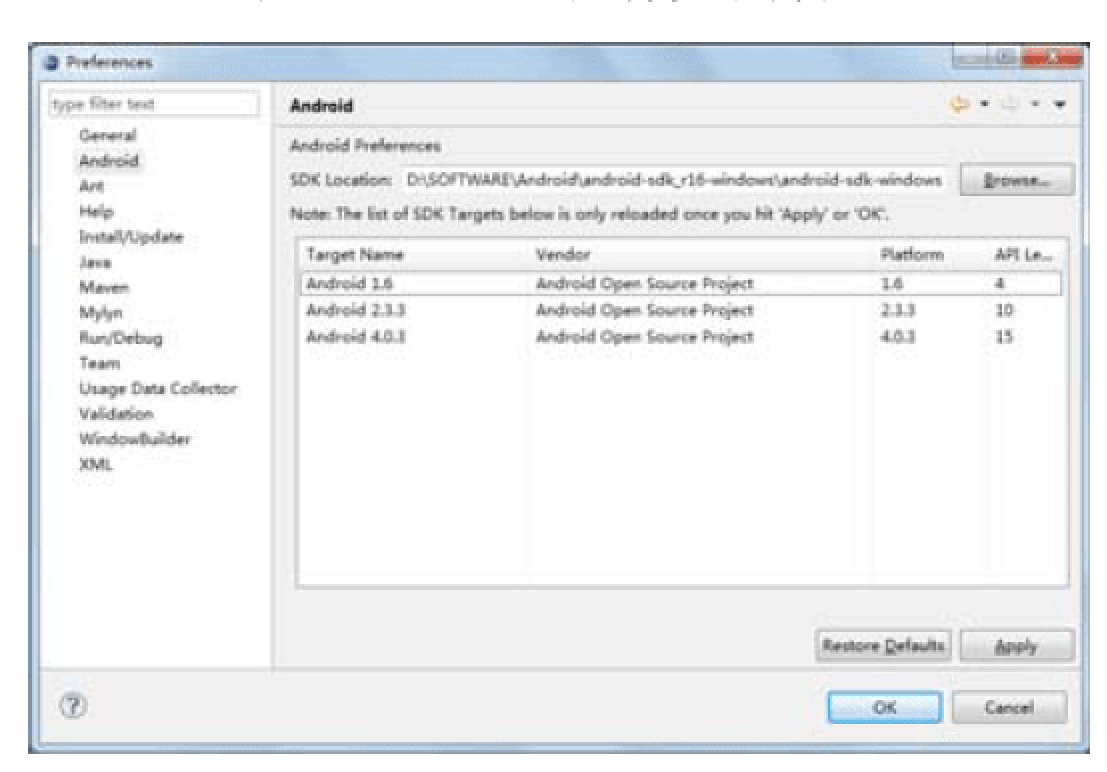


图 2-15 ADT 安装目录选择界面

此时，再次打开该窗口，就可以看到 SDK 列表了。

（5）最后，配置 Eclipse 的 Run Configuration。选择菜单 Run→Run Configurations，双击 Android Application 创建一个新的配置文件，指定右侧 Android 选项卡中的 Project 项目，选择需要的版本，单击 Apply 按钮，如图 2-16 所示。

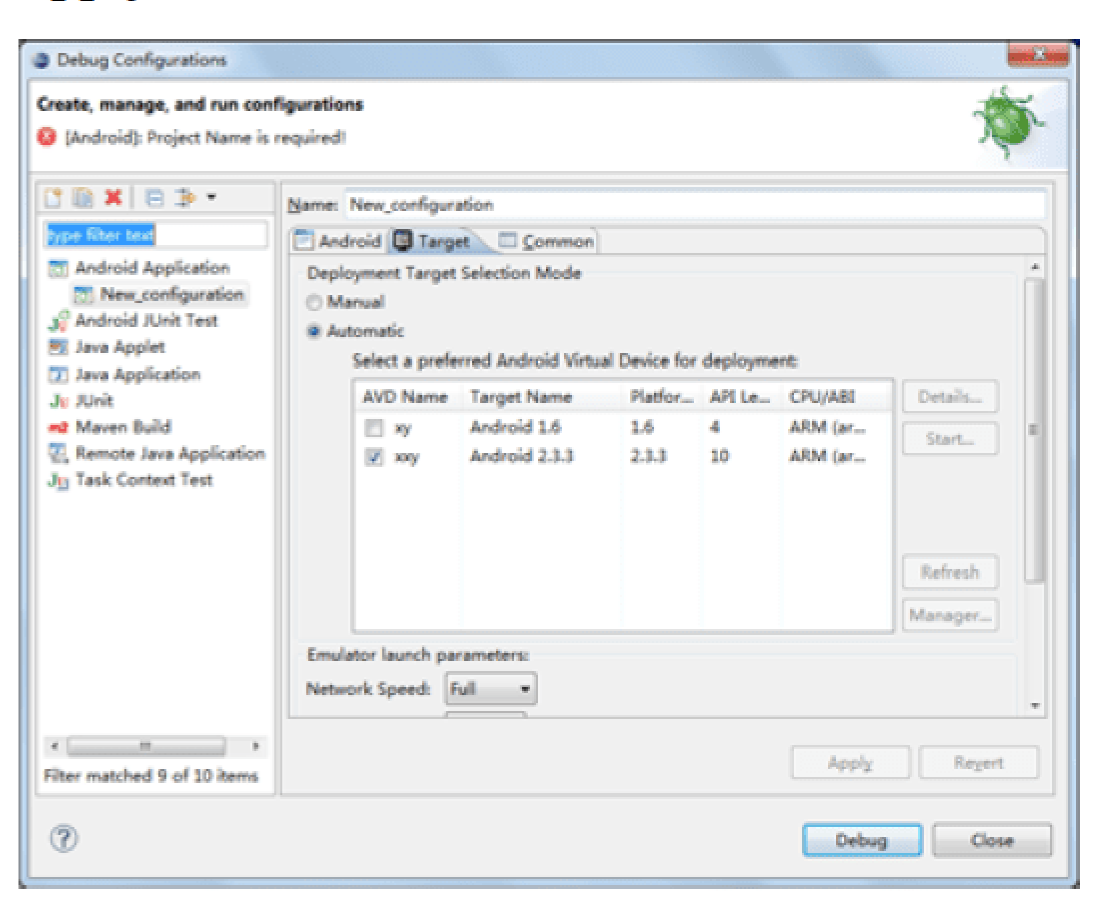
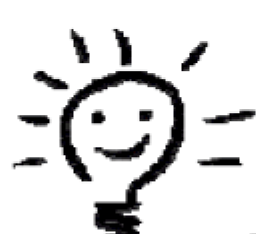
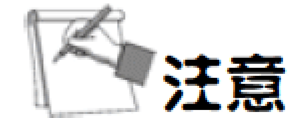


图 2-16 Eclipse 的配置运行设置界面



至此，Eclipse 下的 Android 开发环境已搭建成功。



注意

（1）Windows 7 与 Windows XP 系统的安装过程基本相同。
（2）所有的文件路径需要全英文命名，否则不能安装成功。

任务 4　我的第一个 Android 项目——HelloAndroid

【任务情境】

前面已经对 Android 的开发环境和模拟器进行配置，本任务将带领读者构建第一个 Android 程序并对其进行简单的讲解。

【相关知识】

1. 创建第一个 Android 应用程序

（1）打开 Eclipse，选择菜单 File→New→Project 命令，在打开的 New Project 对话框中单击选择 Android Project 命令，如图 2-17 所示，单击 Next 按钮。

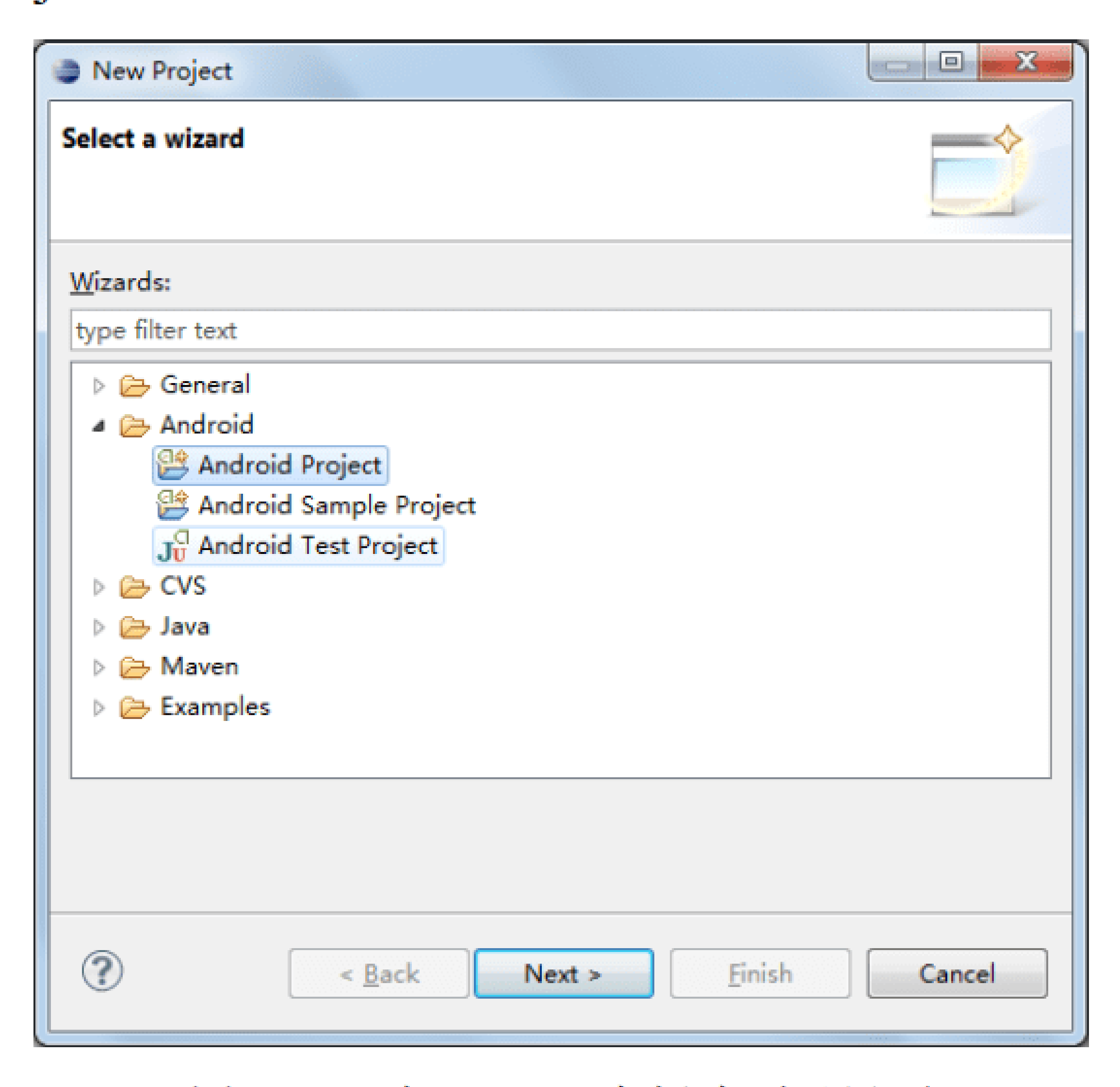


图 2-17　在 Eclipse 中新建项目界面

（2）然后在弹出的窗口中填写 Project Name 为 sample1，单击 Next 按钮；这里选择 Android2.3.3 版本，单击 Next 按钮；参考图 2-18 完成其他基本信息的填写，最后单击 Finish 按钮即可。

（3）成功创建后可在 Package Explorer 窗口中观察到该项目的目录结构，如图 2-19



所示。

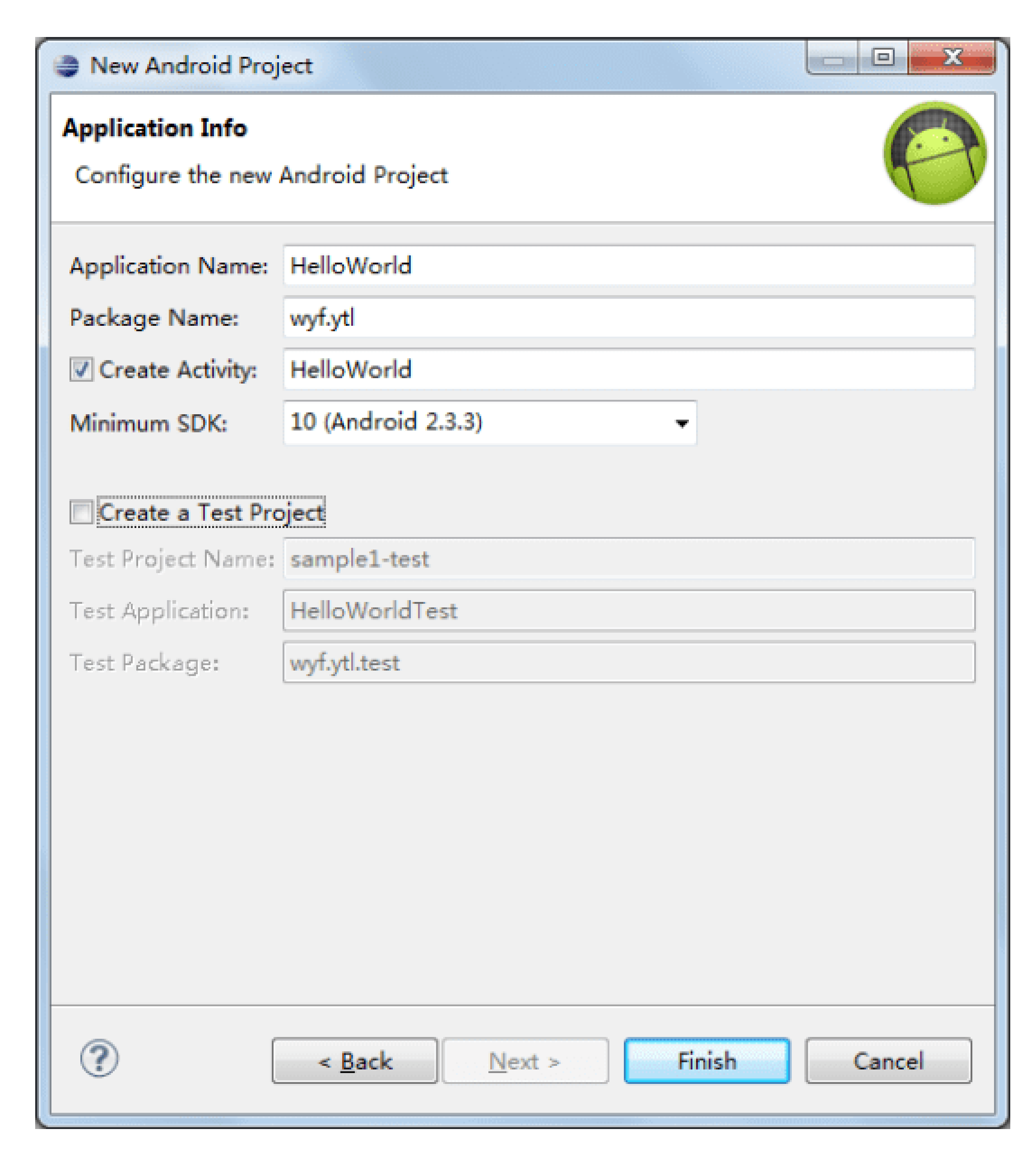


图 2-18　Eclipse 新建项目信息的填写界面

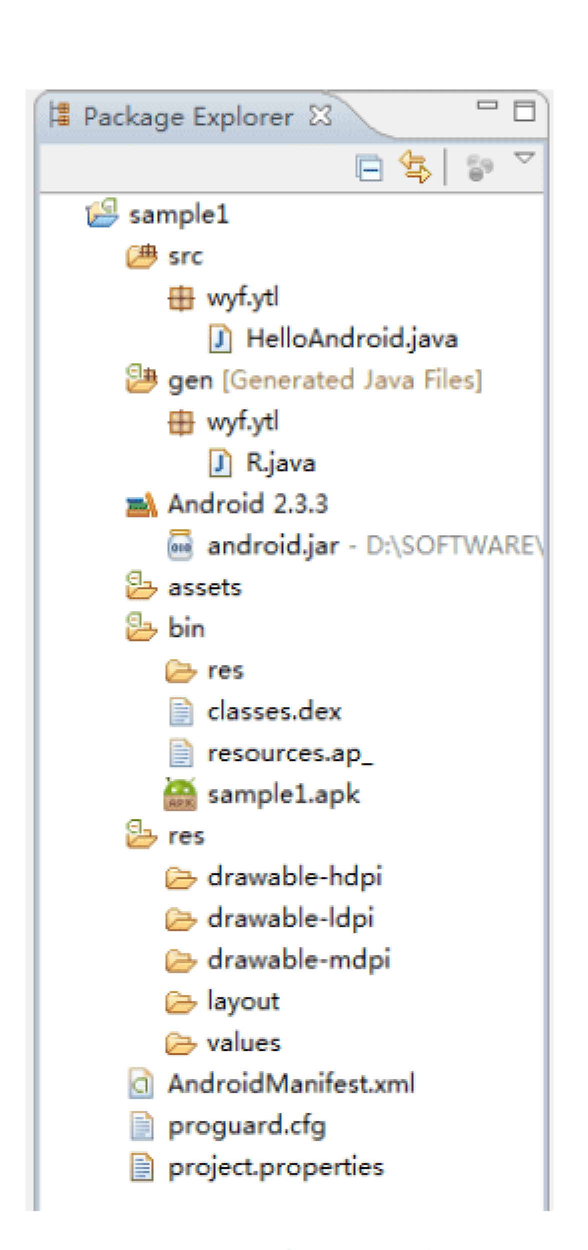


图 2-19　Eclipse 中项目的目录结构

（4）此时，在项目名上单击鼠标右键，选择 Run As→Android Application 命令即可运行刚才创建的 HelloAndroid 项目，运行效果如图 2-20 所示。

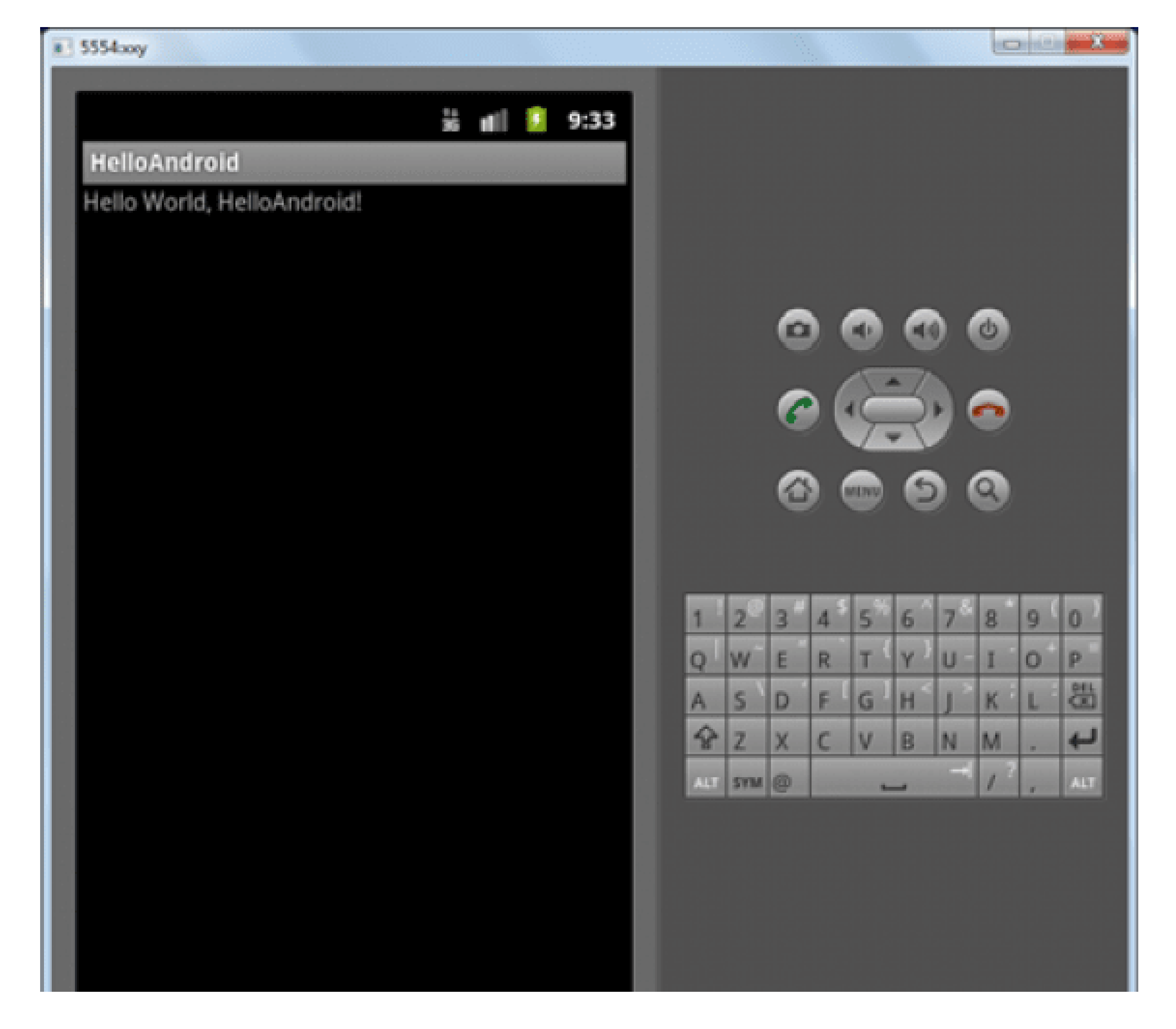


图 2-20　HelloAndroid 项目运行效果



提示

因为有很多程序或者游戏是横屏模式，所以在程序调试过程中，可能需要将模拟器切换成横屏模式，此时可以通过快捷键 Ctrl+F12 来切换模拟器的横、竖屏模式。在横屏模拟器中运行 HelloAndroid 程序的效果如图 2-21 所示。



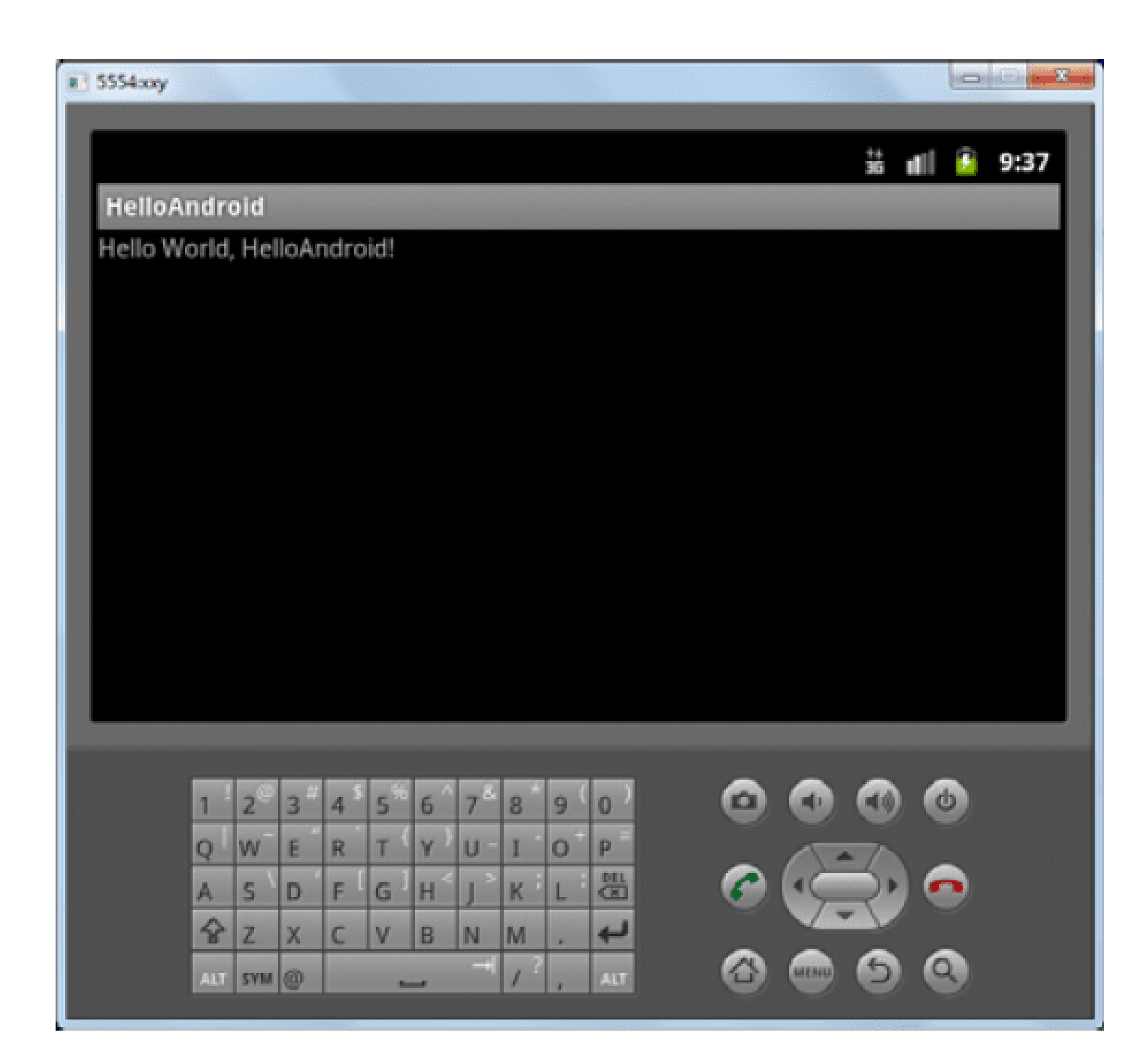


图 2-21　HelloAndroid 项目运行横屏效果

2. HelloAndroid 的简单讲解

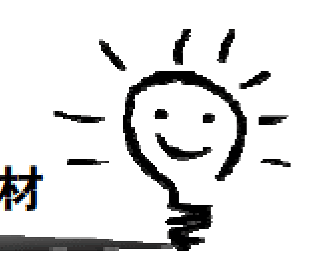
通过前面的讲解，读者已经能够创建并运行简单的 Android 程序了，但可能对 Android 项目还不够了解，接下来将通过对 HelloAndroid 程序的详细介绍，使读者了解 Android 项目的目录结构以及 HelloAndroid 的运行原理。

（1）先来看看 HelloAndroid 项目中各个目录和文件的作用。

- ☑ src 目录。src 目录用来存放应用程序中所有的源代码，代码的源文件一般存放在相应的包下面，在开发 Android 应用程序时，大部分时间都是在编写 src 中的源代码。
- ☑ gen 目录。该目录下一般只有一个文件，即 R 文件。该文件是由 ADT 自动产生的，用来存放应用程序中所使用的全部资源文件的 ID，在应用程序开发过程中只是使用 R 文件，一般不需要人工修改该文件。
- ☑ Android2.3.3 目录。该目录存放的是项目所需要的支持.java 包。
- ☑ assets 目录。该目录存放应用程序中使用的外部资源文件，程序中可以通过输入/输出流对该目录中的文件进行读写。
- ☑ res 资源目录。该目录下有多个目录，分别用来存放程序中用到的图片、界面布局文件及 XML 格式的描述文件。
- ☑ AndroidManifest.xml。该文件是整个程序的系统控制文件，是每个应用程序都不可缺少的，并描述了应用程序的组件、资源和权限等。

（2）AndroidManifest.xml 文件是该项目的系统控制文件，该文件的代码如下。

```
<?xml version="1.0" encoding="utf-8"?>        <!--XML 的版本以及编码方式-->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="wyf.ytl"
  android:versionCode="1"
  android:versionName="1.0">
<!--该标记定义了该项目的使用架设，所在的包以及版本号>
```

```
6    <application android:icon="@drawable/ic_launcher" android:label="@string/app_name" >
7  <!--定义了该项目在手机中的图标以及名称-->
8         <activity android:name=".HelloAndroid"
9               android:label="@string/app_name" > <!--声明 Activity 组件-->
10              <intent-filter>
11                   <action android:name="android.intent.action.MAIN" />
12                   <category android:name="android.intent.category.LAUNCHER" />
13              </intent-filter> <!--声明 Activity 可以接受的 Intent-->
14         </activity>
15    </application>
16    </manifest>
```

☑ 第 1～5 行定义了程序的版本、编码方式、用到的架构以及该程序所在的包与版本号。

☑ 第 6 行定义了程序在手机上的显示图标及显示名称。

☑ 第8～14行定义了一个名为HelloAndroid的Activity以及该Activity能够接受的Intent。

（3）main.xml 是该项目的布局文件，其代码如下。

```
1  <?xml version="1.0" encoding="utf-8"?> <!--XML 的版本以及编码方式-->
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3    android:layout_width="fill_parent"
4    android:layout_height="fill_parent"
5   android:orientation="vertical"
6   ><!--定义了一个线性布局，布局方式是垂直的-->
7    <TextView
8         android:layout_width="fill_parent"
9         android:layout_height="wrap_content"
10        android:text="@string/hello"
11  /><!--向线性布局中添加一个 TextView 控件-->
12  </LinearLayout>
```

☑ 第2～4行定义了布局方式为LinearLayout，且左右和上下的填充方式为fill_parent。

☑ 第 7～11 行中向该布局中添加了一个 TextView 控件，其宽度和高度模式分别为 fill_parent、wrap_content，在 TextView 控件显示的内容为 string.xml 中的 hello 内容。

（4）项目的主类 HelloAndroid.java 的代码如下。

```
1    package wyf.ytl;
2    import android.app.Activity;
3    import android.os.Bundle;
4    public class HelloAndroid extends Activity {
5   @Override
6    public void onCreate(Bundle savedInstanceState) {
7         super.onCreate(savedInstanceState);
8         setContentView(R.layout.main);
9      }
10   }
```

☑ 第 4 行是对继承 Activity 子类的声明。



☑ 第 6～9 行重写了 Activity 的 onCreate 回调方法，在 onCreate 方法中先调用基类的 onCreate 方法，然后指定用户界面为 R.layout.main，对应的文件为 res\layout\main.xml。

任务 5　Android 程序的监控与调试

【任务情境】

前面几个任务已经对 Android 应用程序的创建进行了详细讲解，本任务将介绍如何通过 DDMS 来监控 Android 应用程序的运行以及如何调试 Android 程序。

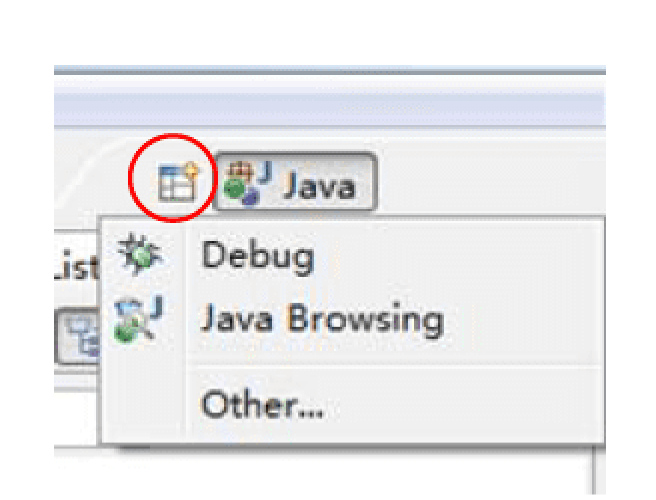
【相关知识】

本任务在调试过程中使用了 android.util.log 类，该类简单易用。监控与调试的详细步骤如下。

（1）打开刚刚创建的项目，找到 HelloAndroid.java 文件，在第 7 行 “super.onCreate (savedinstanceState);” 之后添加 “Log.d(“TAG”,”This is message!”);” 语句。

（2）在项目名上单击鼠标右键，然后选择 Run As→Android Application 命令运行该项目。

（3）依次单击 Eclipse 右上角的 Open Perspective→Other。Eclipse 将弹出 Open Perspective 对话框，如图 2-22 所示。



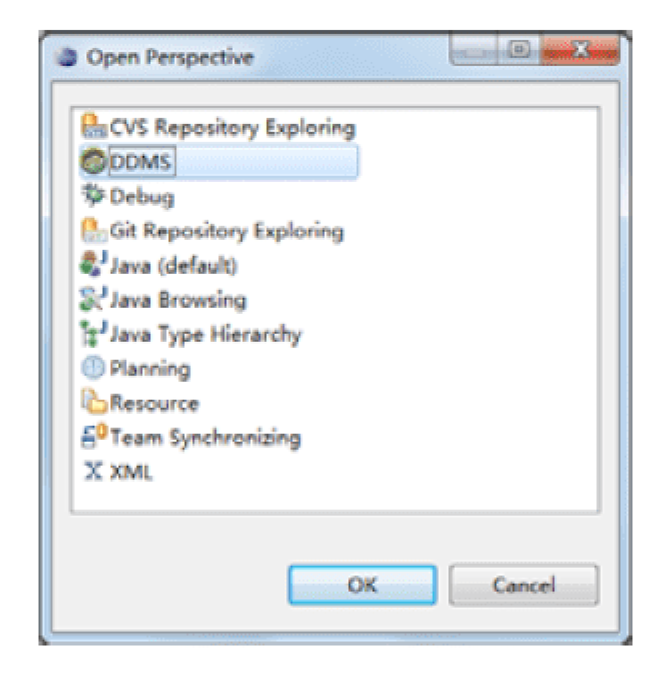


图 2-22　HelloAndroid 项目调试界面

（4）单击 OK 按钮后将 Eclipse 切换到 DDMS 视角，如图 2-23 所示。LogCat 显示在屏幕的下方，系统中所有的日志都将出现在 LogCat 中，通过对 LogCat 的观察可以详细了解 Android 程序运行的过程。



在程序的开发和调试过程中，少不了对文本的输出，而在 Android 程序的开发中，建议使用 Log 类来打印需要打印的文本。



（5）在图 2-23 中可以看到程序中添加的日志输出，这样在程序的开发过程中可以随时使用 Log 类来打印需要打印的信息，而当 LogCat 中日志过多时，可以使用过滤器 Filter 对 Tag 进行过滤来筛选日志。

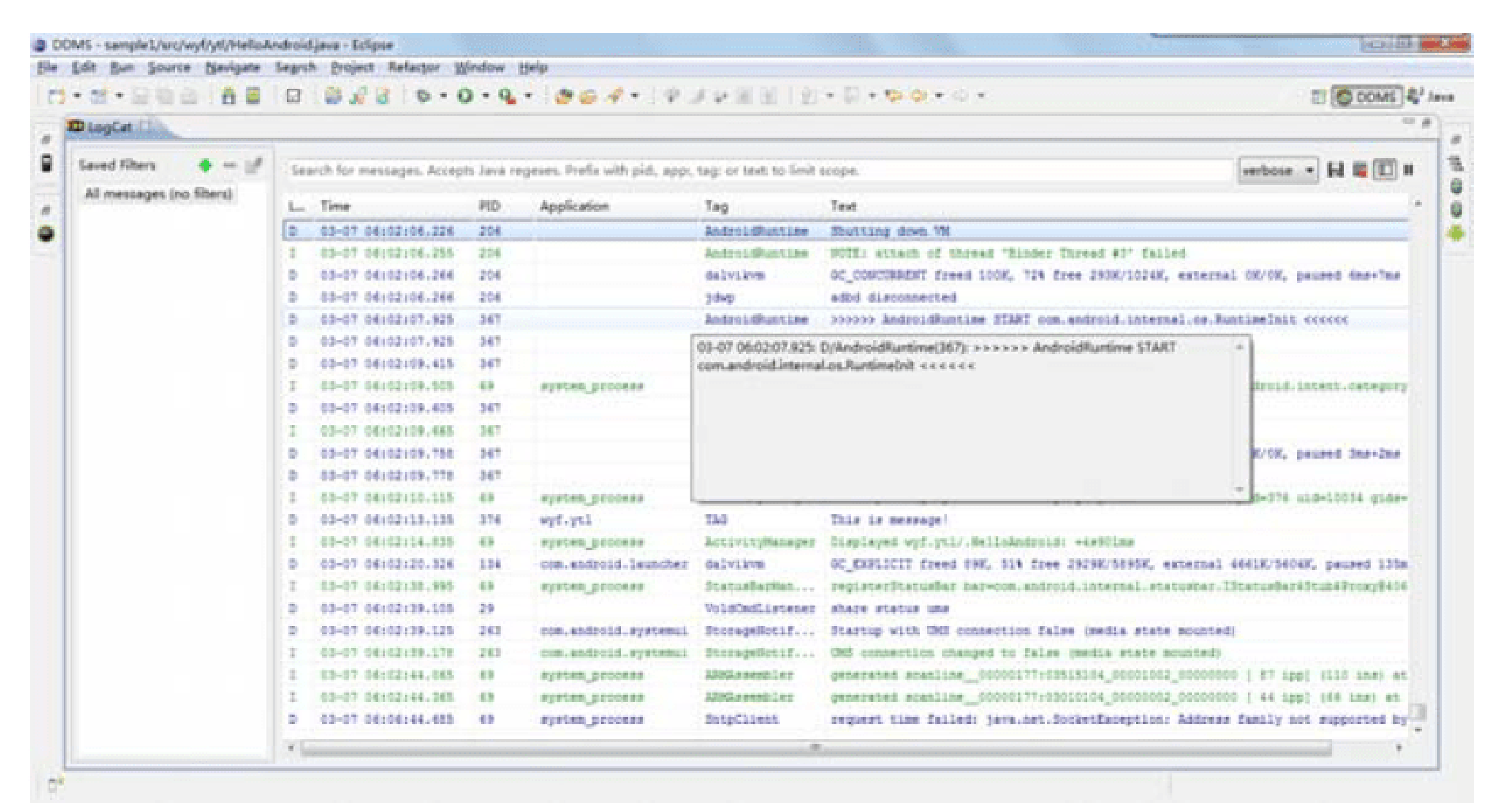


图 2-23 HelloAndroid 程序运行结果界面

【项目小结】

本项目介绍了 Android 平台的来源及优点，并详细介绍了在 Eclipse 中如何构建 Android 的开发环境，最后创建了一个 Android 的应用程序。通过本项目的学习，读者应该已经对 Android 平台下应用程序的开发步骤有了初步的了解。

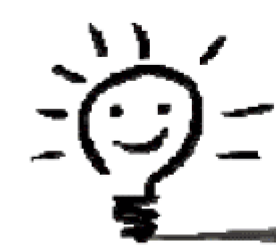
综合实训二 Android 游戏发展的未来

由于一些具有影响力的商业开发公司加入 Android 游戏开发，这必将刺激越来越多的开发者对 Android 游戏的兴趣，从而使 Android 迎来一个游戏井喷期。但是，事实上光明的前途往往伴随着曲折的道路。如果我们乐观地认为 Android 游戏的发展从此将一帆风顺、高歌猛进，那我们可能会失望。

Android 游戏之所以发展缓慢，不仅仅是因为 Google 在游戏和娱乐上固有的短板。Android 版本众多，各个手机厂商深度定制导致代码的分裂以及差别巨大的硬件配置等都成为 Android 游戏发展的阻碍。

而事实上，Android 游戏发展面临的真正的阻力却来自 Google 自身，或者说，从 Android 诞生的那一刻便已经决定了。

一方面，由于 Android 是开源的，理论上任何厂商都可以根据自己的需求对其进行定制，于是出现了 HTC Sense、MOTOBLUR 等各种定制 UI，甚至出现了移动 OMS、联想乐 phone 等基于 Android 的系统。定制 UI 一方面提升了手机界面的差异性，也对企业塑造其品牌形象大有帮助，但是它却导致 Android 代码分裂，应用程序彼此难以兼容，开发者的开发成



本大幅度提高。

Android 诞生短短两年时间，系统版本便经历了几次大的变革，版本更新是好事，说明 Android 正在朝着越来越好用、越来越完善的方向发展，但是有一个致命的问题便是针对新的版本开发的应用往往无法在旧的版本上运行。于是同一个应用，开发者往往要针对不同的系统版本开发不同应用，再加上 Android 手机在硬件上不统一，导致开发难度进一步提升，使得很多游戏开发者不愿意在 Android 上做过多的努力。

Google 不是苹果，苹果牢牢地控制着 iPhone 的系统版本和硬件，高度统一的系统和硬件为开发者提供了一套完整的开发规范，再加上苹果对 App Store 的严格审核，游戏开发者只要按照开发规范开发，通过审核并在 App Store 上架就可以准备推广然后在家数钱了。但是 Google 不一样，Android 采用的是开放手机联盟的方式，Google 无法规定厂商采用何种硬件和哪个版本的 Android 系统，开源一方面成就了 Android，另一方面也带来了 Android 世界的混乱，导致应用开发难度和推广难度的大幅度提高。

但是，事实上 Google 已经在努力改变这一切。同时由于 Android 的快速发展，Android 游戏市场这块巨大的蛋糕必然引来更多游戏开发者的加入。只是，道路依然曲折。Google 近日放出消息，Android 3.0 禁止厂商自定义界面，力图挽回 Android 世界的混乱局面。

第二部分

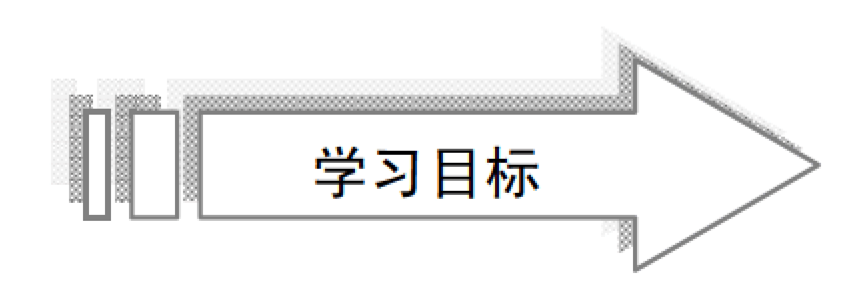
高级篇——游戏综合实现

项目三 战国英雄传游戏介绍

项目情境

“战国英雄传”是一款典型的策略游戏，允许玩家自由控制、管理和使用游戏中的人物、资源，并通过较为自由的手段对抗敌人以达到游戏所要求的目标，取得游戏的胜利。“战国英雄传”取材于战国时期诸侯纷争的年代，游戏中玩家通过掷骰子控制主人公的行走，并通过合理地管理自己的城池与将领、合理地分配粮草与兵力来达到统一中原的目标。

我们通过介绍策略类游戏——“战国英雄传”在 Android 平台上的设计与实现，使读者了解该类游戏的开发过程，同时掌握大型 RPG 游戏常用的设计模式与开发思路。



- 了解“战国英雄传”的游戏背景和实现的功能
- 熟悉“战国英雄传”游戏的开发环境和目标平台

任务6　游戏背景及功能概述

【任务情境】

本任务将对该游戏的背景以及所要实现的功能进行简单介绍，使读者在正式进入开发之前对该游戏有一定的感性认识。

【相关知识】

1. 背景概述

早期的策略游戏玩法比较单一，游戏结果一般是统一国家或开拓殖民地，后来逐步发展成游戏玩法比较固定的模拟类游戏。在游戏中通过模拟现实生活中或过去的世界，充分利用自己的智慧来建立城池、招募将领，并通过努力管理城池和将领以达到游戏所设计的目标。

在“战国英雄传”游戏中，玩家可以合理地管理兵力、粮草等资源，并通过攻占敌方城池、开疆拓土来完成最终的统一中原大业。

2. 功能简介

下面对该游戏基本功能以及程序的运行步骤进行简单介绍。

（1）运行该游戏，首先出现的是简单的声音询问界面，询问玩家是否需要开启游戏。

（2）之后将进入如图3-1所示的加载界面，此时后台正在加载菜单界面的资源。

（3）加载完毕后将进入主菜单界面，如图3-2所示，并根据设置播放背景音乐。玩家可以通过“音效设置”菜单进入音效设置界面，在音效设置界面中可以对背景音乐、开场音乐、战斗音乐以及环境音效进行设置。



图3-1　游戏加载界面



图3-2　游戏主菜单界面



（4）如果在菜单界面，玩家单击“江湖指引”菜单将进入帮助界面，如图 3-3 所示。

（5）而如果在菜单界面单击“群英荟萃”菜单便会进入关于界面，显示游戏的相关信息，其效果如图 3-4 所示。



图 3-3　游戏帮助界面



图 3-4　游戏信息界面

（6）如果之前保存过游戏，则可通过“风云再起”菜单直接进入游戏界面；反之可通过“初入江湖”菜单开始新的游戏，首先进入游戏背景介绍界面，之后会进入游戏界面，如图 3-5 和图 3-6 所示。



图 3-5　游戏背景介绍界面

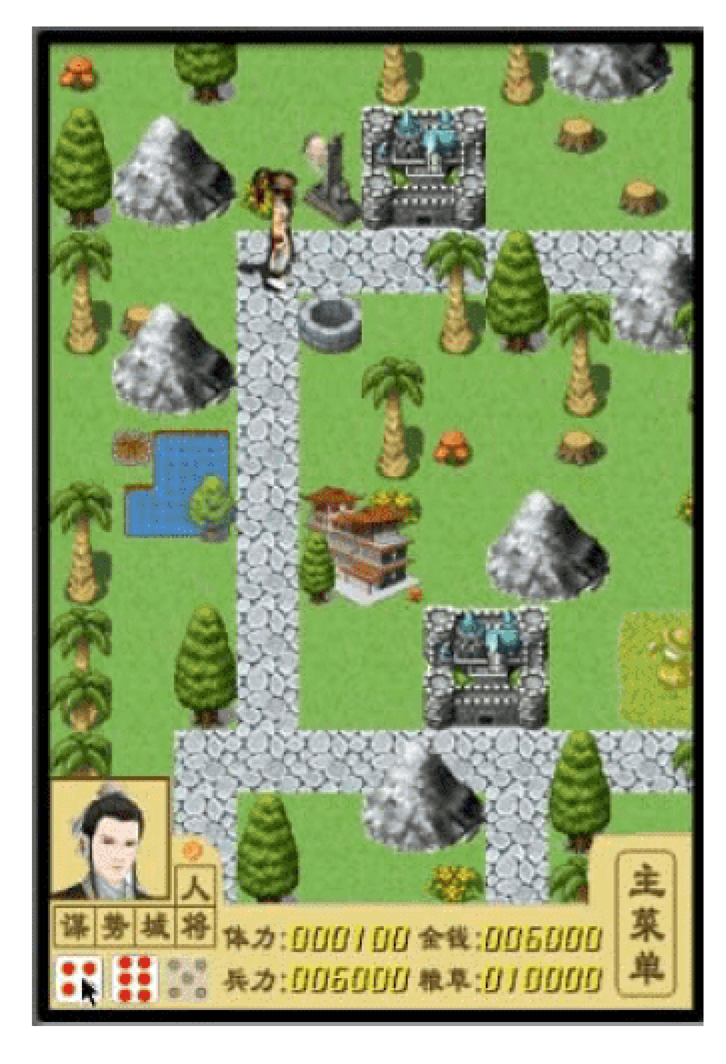


图 3-6　游戏界面

（7）在游戏界面中，可单击“主菜单”按钮进入主菜单，在主菜单界面中可载入之前的存档，也可以保存当前的存档，还可以查看帮助或退出游戏，效果如图 3-7 所示。

（8）游戏的玩法是通过单击人物头像掷骰子，然后根据所掷的点数来移动人物。

（9）单击“人”按钮可查看自身属性、技能等信息，如图 3-8 所示，可通过右上角的翻页按钮来控制页面的显示。同样，单击“将”按钮可查看并管理玩家所拥有的所有将领，

在该界面中还可以对将领执行查看详细信息、任免官职、会面、指派等操作。单击“城”按钮可查看英雄拥有的所有城池。单击“详细”按钮便可对选中的城池进行管理，如分配粮草，调拨兵力、箭垛、战车等。通过“势”按钮可以查看当前天下所有城池的基本信息。在“谋”界面中可以使用当前玩家所能使用的技能，选中需要使用的技能后单击“使用”按钮即可。当遇到敌方城池时，可选择缴税或者攻城，如选择攻城便会出现攻城动画，显示战斗过程，战斗结束后显示战斗结果。



图 3-7　游戏主菜单界面



图 3-8　游戏“个人属性”界面

（10）在游戏过程中遇到麦田或稻田可以得到粮食，遇到村庄可以招兵买马，遇到矿山、渔场、森林可以得到金钱，遇到学堂可以为武将传授知识，遇到驿站可以恢复体力，遇到鲁班可以学习建造箭垛与战场，遇到马场可以训练武将马术。

（11）而当遇到敌方城池却交不起过路费时，游戏则失败，需重新开始游戏；而当统一天下后，游戏胜利。

3. 目标平台

本游戏的目标平台为 Android2.3。

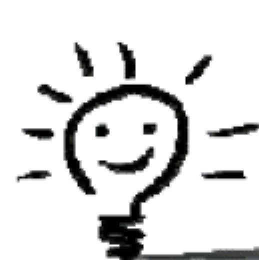
【项目小结】

本项目主要介绍了“战国英雄传”游戏的简要情况，包括游戏背景、游戏功能以及开发该游戏需要的环境和目标平台，在正式进入游戏开发之前，了解这些信息能够加强对游戏的认识，帮助开发进程。在设计并开发一款游戏前，首先要定义好以上背景，为游戏的开发奠定基础，这正是“磨刀不误砍柴工”。

📖 小知识一：什么是策略游戏

☑ **SLG = Simulation Game：策略游戏**

这类游戏提供给玩家一个可以多动脑筋思考问题、处理较复杂事情的环境，允许玩家



自由控制、管理和使用游戏中的人或事物，通过这种自由的手段以及玩家们开动脑筋想出的对抗敌人的办法来达到游戏所要求的目标。

策略游戏可分为回合制和即时制两种。

回合制策略游戏如大家喜欢的《三国志》系列（三国志 7、8、10 为 RPG 类型游戏）、《樱花大战》系列，又如《星战会》和策略类网游《太空帝国》；即时制策略游戏如《命令与征服》系列、《帝国时代》系列和《沙丘》等。

后来在策略类游戏的发展中形成了一种游戏方法比较固定的模拟类游戏。这类游戏主要是通过模拟现实生活在虚拟的环境里经营或建立医院、商店等场景。要充分利用自己的智慧去努力实现游戏中建设和经营这些场景的要求，即 SIM（simulation）类游戏（如《模拟人生》、《模拟城市》、《过山车大亨》、《主题公园》等）和养成类游戏 TCG（如《足球经理世界》等），还有以培养和教育游戏对象为目的的养成类游戏和即时策略类游戏，也都是策略类游戏的发展和分支，如《零波丽育成计划》等也归到了 SLG 下。

除策略类游戏外，主要还有以下几类：

☑ **RPG=Role-playing Game：角色扮演游戏**

由玩家扮演游戏中的一个或数个角色，有完整的故事情节的游戏。玩家可能会与冒险类游戏混淆，其实区分很简单，RPG 游戏更强调的是剧情发展和个人体验，一般来说，RPG 可分为日式和美式两种，主要区别在于文化背景和战斗方式。日式 RPG 多采用回合制或半即时制战斗，如《最终幻想》系列，大多国产中文 RPG 也可归为日式 RPG，如大家熟悉的《仙剑》系列、《剑侠》系列等；美式 RPG，如《暗黑破坏神》系列。

☑ **ACT=Action Game：动作游戏**

玩家控制游戏人物用各种武器消灭敌人以过关的游戏，不追求故事情节，如熟悉的《超级玛丽》、可爱的《星之卡比》、华丽的《波斯王子》等。电脑上的动作游戏大多来自早期的街机游戏和动作游戏，如《魂斗罗》、《三国志》等，设计主旨是面向普通玩家，以纯粹的娱乐休闲为目的，一般有少部分简单的解谜成分，操作简单、易于上手、紧张刺激，属于大众化游戏。

☑ **AVG=Adventure Game：冒险游戏**

由玩家控制游戏人物进行虚拟冒险的游戏。与 RPG 不同的是，AVG 的特色是故事情节往往是以完成一个任务或解开某些迷题的形式出现的，而且在游戏过程中刻意强调谜题的重要性。AVG 也可再细分为动作类和解迷类两种：动作类 AVG 可以包含一些格斗或射击成分，如《生化危机》系列、《古墓丽影》系列、《恐龙危机》等；而解迷类 AVG 则纯粹依靠解谜拉动剧情的发展，难度系数较大，如《神秘岛》系列。

☑ **RTS=Real-Time Strategy Game：即时战略游戏**

RTS 本来属于策略游戏 SLG 的一个分支，但由于其在世界上的迅速风靡，使之慢慢发展成了一个单独的类型，知名度甚至超过了 SLG，有点像现在国际足联和国际奥委会的关系。代表作有《魔兽争霸》系列、《帝国时代》系列、《星际争霸》等。后来，从其上又衍生出了所谓“即时战术游戏”，多以控制一个小队完成任务的方式，突出战术的作用，以《盟军敢死队》为代表。



☑　**FTG=Fighting Game：格斗游戏**

由玩家操纵各种角色与电脑或另一玩家所控制的角色进行格斗的游戏。按技术可再分为 2D 和 3D 两种：2D 格斗游戏有著名的《街霸》系列、《侍魂》系列、《拳皇》系列等；3D 格斗游戏如《铁拳》、《高达格斗》等。此类游戏谈不上什么剧情，最多有个简单的场景设定或背景展示，场景、人物、操控等也比较单一，但操作难度较大，主要依靠玩家迅速的判断和微操作取胜。

☑　**STG= Shooting Game：射击类游戏**

这里所说的射击类，并非是类似《VR 特警》的模拟射击（枪战），而是指纯的飞机射击，由玩家控制各种飞行物（主要是飞机）完成任务或过关的游戏。此类游戏分为两种：一种叫科幻飞行模拟游戏（Science-Simulation Game），非现实的，以想象空间为内容，如《自由空间》、《星球大战》系列等；另一种叫真实飞行模拟游戏（Real- Simulation Game），以现实世界为基础，以真实性取胜，追求拟真，达到身临其境的感觉，如《Lockon》系列、《DCS》、《苏-27》等。

☑　**FPS=First Personal Shooting Game：第一人称视角射击游戏**

严格来说它是属于动作游戏的一个分支，但和 RTS 一样，由于其在世界上的迅速风靡，使之发展成了一个单独的类型，典型的有《使命召唤》系列、《DOOM》系列、《QUAKE》系列、《虚幻》、《半条命》、《CS》……。

☑　**PZL=Puzzle Game：益智类游戏**

Puzzle 的原意是指以前用来培养儿童智力的拼图游戏，引申为各类有趣的益智游戏，总的来说适合休闲，最经典的就是大家耳熟能详的《俄罗斯方块》。

☑　**体育竞技类游戏**

主要是模拟各种体育赛事的游戏，比如《实况足球》等。

☑　**RCG=Racing Game：竞速游戏（也有称作为 RAC 的）**

在电脑上模拟各类赛车运动的游戏，通常是在比赛场景下进行，非常讲究图像音效技术，往往是代表电脑游戏的尖端技术。惊险刺激，真实感强，深受车迷喜爱，代表作有《极品飞车》、《山脊赛车》、《摩托英豪》等。RCG 又可称之为 Driving Game。目前，RCG 内涵越来越丰富，出现了一些其他模式的竞速游戏，如赛艇、赛马等。

☑　**CAG=Card Game：卡片游戏**

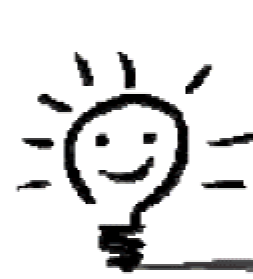
玩家操纵角色通过卡片战斗模式来进行的游戏称为卡片游戏。丰富的卡片种类使得游戏富于多变化性，给玩家带来无限的乐趣，如著名的《信长的野望》系列、《游戏王》系列，还包括卡片网游《武侠 Online》，从广义上说《王国之心》也可以归于此类。

☑　**TAB=Table Game：桌面游戏**

顾名思义，从以前的桌面游戏脱胎到电脑上的游戏，如各类强手棋（即掷骰子决定移动格数的游戏），经典的有《大富翁》系列；棋牌类游戏也属于 TAB，如《拖拉机》、《红心大战》、《麻将》等。

☑　**MSC=Music Game：音乐游戏**

此类游戏可培养玩家音乐敏感性、增强音乐感知。伴随美妙的音乐，有的要求玩家翩翩起舞，有的要求玩家手指体操，如大家都熟悉的跳舞机，就是个典型，目前的人气网游



《劲乐团》也属其列。

☑ **WAG=Wap Game：手机游戏**

目前可以随处玩游戏，连手机也必带休闲游戏，网民最喜欢手机游戏的类型中，益智类比率最高，其次为动作类、战略类、模拟类、射击类等，如《金属咆哮》、《FF7 前传》等。

☑ **MUD=Multiple User Domain：泥巴游戏**

主要是指依靠文字进行的游戏，图形作为辅助。1978 年，英国埃塞克斯大学的罗伊・特鲁布肖用 DEC-10 编写了世界上第一款 MUD 游戏——MUD1，它是第一款真正意义上的实时多人交互网络游戏，这是一个纯文字的多人游戏世界。代表作有《侠客行》、《子午线 59》、《万王之王》等。

综合实训三　微博随身之概况介绍

【问题情境】

本实训将简单介绍社交分享平台——微博随身的系统背景，并对其 Web 端和 Android 手机端的功能结构进行简单说明，使读者对整个系统有一个整体的认识。

【拓展知识】

1．背景介绍

目前 Web2.0 是越来越被人们所提及的热门话题，Web2.0 时代的一个最主要的特征就是每个参与的人既是内容的读者，也是内容的提供者。博客是 Web2.0 时代最具标志性的一项网络服务。在未来的网络中，以人为本的博客将会受到更多的青睐。

随着移动网络技术的推进，风靡全球的博客也开始向移动化和简洁化发展，于是产生了微型博客这种更便捷的信息发布和共享平台。本实训将要介绍的微博随身便是微型博客的简单实现，其中包括 Web 服务器及手机平台中服务端和客户端。

2．功能概述

本系统主要为用户提供一个信息发布和共享平台，用户通过浏览器访问 Web 服务器和通过 Android 手机端访问服务器时享受到的功能基本相同，主要包括如下几点。

☑ 用户注册，为初次使用本系统的用户提供注册服务。

☑ 用户登录，让已注册用户登录平台。

☑ 更新心情，用户可以更新自己的心情，该心情对其他好友是可见的。

☑ 发表日志，用户可以发布新的日志，并可以对发表过的日志进行编辑或删除。

☑ 对登录用户提供上传图片的功能，上传的图片可以用做个人相册，也可以用作供所有用户使用的头像。

☑ 管理相册，可以创建新的相册，并向已有的相册上传图片，同时还可以对已有相册进行不同的访问权限设置。

☑ 搜索用户，用户可以通过昵称简单地搜索其他用户，并且可以将搜索到的陌生用户添加为自己的好友。

☑ 查看好友列表和最近访客，用户可以访问自己的好友或访问过自己的人，也可以通过搜索来访问更多的微博用户，进而访问这些用户。

☑ 拍照上传，该功能为 Android 手机端的特有功能，用户可以在应用程序中调用手机的照相机程序拍摄并将其上传到服务器中用户的指定相册。

根据以上功能概述，可以得出系统的功能结构图，其中 Web 端的功能结构如图 3-9 所示，Android 手机端的功能结构如图 3-10 所示。

随身微博 Web 端功能概述

- 用户登录
- 用户注册
- 修改个人资料
- 发表日志
- 管理日志
- 上传头像和图片
- 管理相册
- 搜索用户
- 查看好友和访客

图 3-9　Web 端的功能结构图

微博随身 Android 端功能概述

- 用户登录
- 用户注册
- 发表日志
- 管理日志
- 拍照上传
- 管理相册权限
- 查看好友和访客
- 搜索并访问用户

图 3-10　Android 手机端功能结构图

3. 开发环境和目标平台

下面将对该系统的开发环境进行简单介绍。

（1）开发环境

开发此系统所需要用到的软件环境如下。

- ☑ JDK1.6 及其以上版本，该版本是当前的最高版本，具有很多老版本不具备的新特性。
- ☑ Web 应用服务器，Tomcat6.0 及其以上版本，建议读者选择绿色解压版，便于安装和环境的搭建，该软件可从网上免费下载。
- ☑ 数据库，MySQL5.1 及其以上版本，MySQL 功能强大且安装方便，更增强了数据的完整性及安全性。MySQL 可以从官方网站免费下载。
- ☑ 集成开发环境，Eclipse Java EE IDE for Web Developers 3.5 及其以上版本，该版 Eclipse 支持 EE 系统的开发，且可以从 Eclipse 的官方网站免费下载。
- ☑ Android SDK 及其 Eclipse 开发插件 ADK，二者均可以免费下载，在第 2 章已经对这两个工具的下载进行了说明。

（2）目标平台

- ☑ 客户端浏览器，建议使用 IE 6.0 以上版本。
- ☑ 手机平台为 Android 2.1。

📖 小知识二：MSNS 简介

当今，移动（mobile）和社交（SNS）已经成为互联网中非常火爆的概念，而将两者结合起来，就形成了 MSNS——移动社交网络。

提到 MSNS 的概念，很多时候都被简单地看作是通过移动终端登录自己的社交账户。实际上，MSNS 并非如此简单，表 3-1 列出了一些常见的 MSNS 系统。

表 3-1　MSNS 系统

Systems（系统）	Services（服务）	Architecture（结构）	Purely mobile of Hybrid（纯移动混合）	Location awareness（定位意识）	Interaction method（相互作用方式）
Dodgeball	location notification, message broadcast	S/C, no API available	hybrid	user notification	SMS
Twitter	micro-blogging	S/C, API available	hybrid	user notification	SMS
Jaiku	micro-blogging	S/C, API available	hybrid	user notification	SMS, client software

在表 3-1 中可以看到，现有的 MSNS 平台，都不是单纯意义上的移动终端，而是基于 Web 平台不断发展，如 Dodgeball、Twitter、Jaiku，都是通过手机终端作为业务的接入设备，提供与移动紧密结合的位置服务与传统的移动通信业务相结合，如 SMS（短信业务）。但是同时可以看出，Dodgeball、Twitter、Jaiku 能提供的业务能力只有一些位置服务以及微博客服务，从这个意义上来看，这三个平台还只是个消息中心，并不能像 facebook 的 SNS 平台那样做到丰富的应用。实际上终端需要做的是一些终端应该处理的任务，这些任务并不复杂，比如调用终端固有的功能模块，将信息以表现形式丰富的方式准确、高效地呈献给终端用户，同时服务端除了能提供一些基本的业务，也能提供丰富的、可扩展的业务。

基于现有的 MSNS 平台，我们可以从多个方面来对 MSNS 的含义进行讨论。

从字面上理解 MSNS 是指 Mobile SNS，也就是 Moible+SNS，需要社交服务一定要带



有移动特性，要充分利用移动终端的功能，即如何才能体现移动特征是整个问题的核心。另外，由于移动终端的处理能力有限，因此还需要将系统进行相应的微小化处理，这与传统的 PC 处理是有区别的。

从概念上来看，由于 MSNS 需要用到移动终端，而移动终端的移动性和随身性使得人们的社交活动不仅仅局限于虚拟的 Web 页面，并且越来越趋于现实，使人们的交互不仅仅是虚拟的信息交互，而是面对面的真人交互。MSNS 除了需要将关系网络图像化、形象化、具体化，同时也要将社交网络体现在真实的现实生活中。

从体系和软件设计的角度来看，MSNS 应该针对移动特性去设计，移动终端具有随身性、位置变换性、体积小、屏幕受限、处理性能受限、存储能力受限以及越来越丰富的三维功能等特性，这些都是在设计的时候需要考虑的问题，最终的目标是 MSNS 平台如何提供出丰富的、有价值的、良好用户体验的、较强的用户黏度的移动社交服务，同时让这样的服务越来越多，应用越来越广泛。从这种角度上来看，MSNS 不一定就是需要将 SNS 微小化，很有可能是越来越丰富和庞大，而且由于移动终端的参与，SNS 本身就会越来越丰富。而需要微小化指的是给终端做的事情要尽量“小而精”、“轻而便捷”，而服务端是模块松耦合的，因此能够较方便地提供功能丰富的服务。

另外，从技术层面来看，SNS 的发展如火如荼，它可以提供许多基于社交和行为交互的小应用，另外不可忽略的是 Web 技术带来了轻量级的开发和部署架构，使得开发和部署更加方便。互联网上丰富的 Web Serivce 资源使得 mashup 得到了很好的应用，作为 SNS 平台只需提供一些基本的接口和核心组件就行。但是 MSNS 可能就不一样了，没有这么轻量级而功能强大的技术在终端上广泛使用，symbian 和 J2ME 都过于复杂，掌握周期比较长，开发者并不适应，不适合做比较轻量级的应用，而利用手机浏览器这种“瘦客户端”又无法发挥终端的固有功能，那么移动 widget 和 Flashlite 这样的技术从技术层面上就给 MSNS 业务带来很好的契机，使得终端既能够充分调用手机本地的功能又能够聚合互联网上的资源，真正做到终端和网络的 mashup。

最后，从业务设计的角度来看，需要关注下面两点：第一点，基于移动环境和手机平台的特点，设计移动情境下“诱惑客户产生行为的各种小模块”；第二点，手机平台上呈现每个用户的“空间”，使得更加适合移动终端的特性。在“行为管理”方面，结合手机的特点，寻求折中。

目前国内的 SNS 网站绝大部分都属于互联网 SNS 网站，如果通过手机本身的功能打开 SNS 网站，如校内网，那么除了可以看到基本的好友状态更改、日志分享及更新状态，视频、游戏及其他功能均无法使用。这种必须依托互联网才能实现浏览的限制不利于用户黏度的维护。而 3G 条件下的 MSNS 则可以打破电脑的限制，它的网络传输速度能够使许多功能在手机上实现，弥补 2G 手机网速过慢、很多功能只能通过电脑实现等不足，使 SNS 成为能够随时追踪用户动态的社区网络。MSNS 能够满足用户随时浏览 SNS 网站的需求，并拥有以下几点传统互联网不具备的优势。

☑　**真实性**

用户只要通过手机接入 SNS，得到授权后在技术上通过调用手机通讯录能将 SNS 用户与其真实身份联系起来，从而更容易地反映和建立真实的社会关系。

☑　**功能多样性**

与传统 SNS 的以 PC 浏览器作为接入媒介不同，作为 MSNS 接入媒介的移动终端（手



机、平板电脑等)，在硬件功能上更加多样化。摄像头已经作为基本配置出现在手机上了，而 GPS 传感器也出现在了大部分智能手机上，这就为增加多媒体功能和 LBS 功能奠定了基础。

☑ **便利性**

MSNS 除了具备传统的 SNS 特点外，最大的优势在于它不受传统互联网的限制，只需一台 3G 手机就可以实现诸多 SNS 网站的使用需求。这一点对于希望随时表达个性、对人群互动联络的年轻人很重要。他们可以随时更新自己的状态，了解好友和群组的最新动态，并且能够随时随地使用 SNS 网站的各项功能，进行游戏，上传图片、音乐及视频甚至随时就时事发表话题进行讨论，实现社交网络与生活的同步更新。

☑ **高黏性**

当前国内传统 SNS 网站运行的一大困难是如何维护用户的黏度，而 MSNS 的推出能够让其从普通的、需互联网络支持的 SNS 网站脱颖而出，提供用户时刻上网的功能，通过便利性增加用户登录及使用几率，完成与本 SNS 网站内部好友的即时沟通和信息交流，从而增大用户对于该社区的依赖性，提高用户黏度，从而为网站创造利润制造有利条件。

☑ **高利润**

MSNS 由于具备了随时随地上网的特性，因此其用户将会加大对该社区的使用频率，尤其是社区游戏将会成为用户打发闲暇时间的好选择。MSNS 的游戏开发可以得到充分利用，延长了游戏生命周期。由于扩大了用户对游戏的使用和依赖性，可以在游戏中嵌入更多有偿使用的装备和单向功能，让愿意在游戏中升级的用户自主选择。由于用户有了更多时间打造自己页面，也可以增加页面的有偿装饰功能，以获得利润，从而打破国内传统 SNS 网站难以通过用户获利的状态。同时其高黏性也增强了嵌入式广告的投放效果，强化了广告的影响力，间接为广告投放商增加了效益。

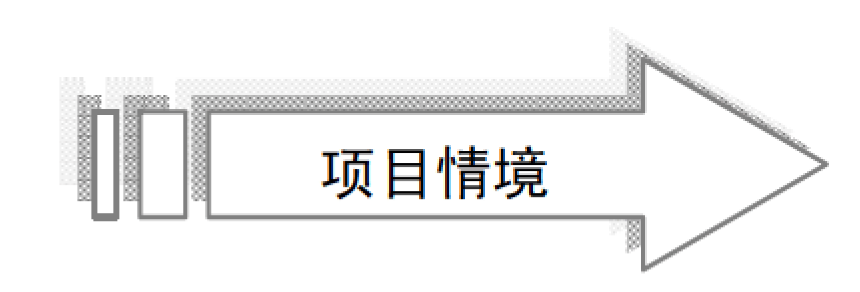
☑ **时尚性**

MSNS 作为 3G 时代的产物，能够充分结合 3G 网络传输速度快以及互联网功能强大的特点，体现其作为 3G 时代新生物小巧但功能完备的特性，得到善于追逐新事物的年轻人的关注。另外，用户可以完全摆脱对传统互联网的依赖，随时随地展示自己的个性，建立自己的圈子，这也是现代年轻人社交的实际需要。而其互动更加快速、简单的特性，也代表了一种新潮、时尚的生活方式，容易受到年轻人的喜爱和推崇。通过将以上优势融合，MSNS 在 3G 的支持下，结合了普通手机的小巧便携性以及传统互联网的沟通性和高传输速度，为用户不受时空限制的登录及使用社区各项功能创造了平台，为网站提供了更多的盈利可能性，也为社区内嵌入式广告供应商的广告投放增强了效果。

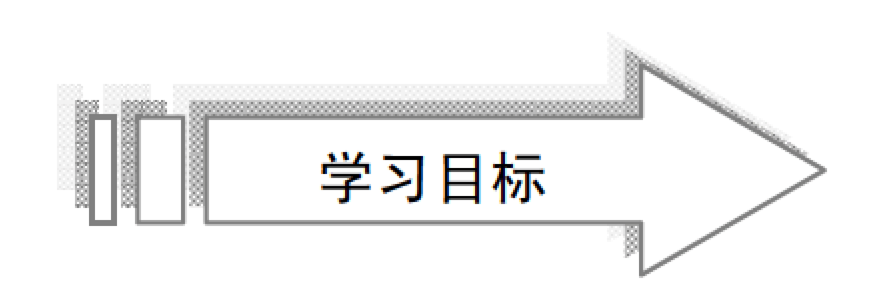
综上所述，关于 MSNS 也可以得出以下结论：

从根本上说，MSNS 除了让用户感受到更个性化和更移动化的服务、体验更丰富和更动态的业务功能外，它还让人与人之间的社交更趋于现实生活，使移动设备真正成用户社交网络和生活方式上一个不可分割的组成部分，MSNS 不仅仅将人与人之间的关系的建立和维护停留在虚拟世界，更让人与人之间的交互不仅仅通过虚拟的社交活动，而是应该让用户有真人互动的感觉，让社交就发生在日常生活当中，让人们面对面的机会更多，最终通过充分运用 SNS 和移动终端的强大能力和特性，提供给用户一个完整的体验：通过移动设备来融合物理世界和虚拟网络的社交交互。

项目四
游戏前热身



了解了“战国英雄传”游戏的背景和功能之后，下面来进行游戏的策划和准备工作，并确定游戏的开发方案。策划是游戏开发周期中必不可少的环节，其中包括故事情节、人物形象等内容的设计以及图片、声音等资源的准备工作。



- 熟悉游戏情节、地图设计器、呈现技术、操作方式等内容
- 搜集和制作游戏所需的资源

任务 7　游戏的策划

【任务情境】

游戏的策划是指对游戏中主要功能的实现方案进行确定的过程，一个大型游戏在开发前需要缜密地策划才可以启动。例如，确定呈现技术、目标平台等内容。

【相关知识】

1. 游戏情节

本游戏的故事背景定在战国乱世，游戏中的人名、城池名、对话等都将根据故事背景来确定。玩家的目标是控制英雄，在群雄并起中通过不断扩张自己的势力最终统一天下。

2. 地图设计器

本游戏的地图界面采用图元技术，由于本游戏中的地图元素不仅仅是通过与否那么简单，因此开发该游戏时必须使用地图设计器，否则在设计地图及地图元素时将很难进行。地图设计器可以使用第三方产品，也可以自己开发。

3. 呈现技术

本游戏采用的游戏视角为正 90、2.5D 俯视视角。同时由于地图的尺寸超过了手机屏幕的尺寸，还需要在游戏中实现滚屏功能。

4. 操作方式

游戏的操作方式为触控操作，在游戏中单击英雄头像掷骰子，英雄会根据骰子点数移动相应的步数。游戏菜单及各种控制面板的弹出也是通过单击屏幕上的按钮来实现的。

任务 8　Android 平台下游戏的准备工作

【任务情境】

本任务进行游戏开发之前的准备工作，主要是搜集和制作游戏中与代码无关的图片和声音资源。

【相关知识】

本游戏中用到的图片资源如表 4-1 所示。



表 4-1　图片资源列表

图　片　名	大小（KB）	像素（w×h）	用　　途
About.jpg	9.41	320×480	关于图片
Battle_field.jpg	5.48	320×480	战场图片
Maitian.png	13.2	62×62	麦田图片
Board.png	11.2	115×53	显示城池
Caodi.png	1.83	31×31	草地图片
Chengshi.png	4.57	62×62	城池图片
Close.png	1.40	60×29	“关闭”按钮
Cunzhuang.png	5.67	62×62	村庄图片
Daotian.png	10.2	62×62	稻田图片
Dark.png	33.0	320×480	遮盖图片
Dashboard.png	4.22	320×116	仪表盘
Dice.png	2.81	150×25	骰子图片
Gonglu.png	1.83	31×31	道路图片
Help.png	1.85	313×312	帮助图片
Hero.png	53.4	62×496	英雄图片
Jing.png	2.5	31×31	井图片
Kuangshan.png	9.47	62×62	矿山图片
Logo.png	2.45	30×30	面板图片
Mozi.png	4.05	31×62	墨子图片
Open.png	1.25	60×29	打开图片
Scroll.png	13.9	280×360	卷轴图片
Senlin.png	4.6	31×62	森林图片
Sunzi.png	3.95	31×62	孙子图片
Wuguan.png	6.96	62×62	武馆图片
Yizhan.png	7.64	62×62	驿站图片
Luban.png	7.2	62×62	鲁班图片
Battle_enemy_general.png	13.6	320×100	敌方将领图片
Dialog_back.png	3.28	320×120	对话框背景
Digit_0～digit_9.png	2.54	10×13	数字图片
Enemy_1～enemy_4.png	6.63	155×80	敌方士兵图片
Game_menu_options.png	8.47	120×240	菜单按钮图片
Head_dynamic.png	3.20	60×60	英雄走路头像
Head_static.png	2.84	60×60	英雄静止头像
Hua1～hua2.png	4.7	31×31	花图片
Hero_soldier_1～4.png	7.67	128×80	己方士兵图片
Battle_hero.png	15.3	328×100	我方将领图片
Loading1～loading2.png	10.9	280×73	加载图片
Zhaoxianguan.png	5.57	62×31	招贤馆图片

续表

图　片　名	大小（KB）	像素（w×h）	用　　途
Machang.png	2.67	62×31	马场图片
Huangcheng.png	8.3	62×62	皇城图片
Menu_options.png	14.3	600×50	菜单按钮文字
Menu_title.png	0.23	300×30	面板标题背景
Menu_back.png	47.6	905×480	菜单背景图片
Buttons.png	6.13	240×30	各种按钮图片
Muzhuang.png	2.21	31×31	木桩图片
Panel_back.png	2.27	320×480	面板背景
Select_back.png	0.2	300×30	选中高亮图片
Sound_back.png	5.59	170×250	声音界面背景
Tiejiangpu.png	8.76	62×62	铁匠铺图片
Xuetang.png	5.59	62×31	学堂图片
Yuchang.png	8.91	62×62	渔场图片

表 4-1 中的图片文件存放在项目文件夹的 res\drawable-mdpi 目录下。下面来准备游戏中用到的声音资源，如表 4-2 所示。

表 4-2　声音资源列表

声 音 文 件	大小（KB）	格　　式	用　　途
Startsound	26.8	mid	菜单界面背景音乐
Battle	23.6	ogg	战斗时播放
Backsound	9.12	mid	游戏进行时背景音乐
Dingdong	10.6	ogg	遇到可遇物时播放

在项目文件夹的 res 目录下新建一个名为 raw 的目录，将表 4-2 所列的声音文件放置到新建的 raw 目录中。

【项目小结】

本项目是帮助同学进行游戏的策划和准备过程，在了解了游戏的背景之后，完善的策划是必不可少的。一份优秀的策划案可以有效地减少开发阶段的工作量，而完整地准备好游戏所需的资源能够保证游戏开发过程顺利进行。

小知识三：Photoshop 开门 10 件事

Adobe Photoshop 是目前最流行的平面设计软件之一。可以说，只要你接触平面设计，你都要和它打交道。关于 Photoshop，要说的实在太多太多，但不论你想让它成为你的左膀右臂，还是仅仅用它来做一些最基础的图像处理工作，那么下面的 10 件事都是你一定要知道的。

（1）快捷键的使用。这是 Photoshop 基础中的基础，也是提高工作效率的最佳方法。



快捷键的使用，就可以将精力更好地集中在作品而不是工具面板上。一旦能够熟练使用快捷键，就可以使用全屏的工作方式，省却了不必要的面板位置，使视野更开阔，能最大限度地利用屏幕空间。



注意

在工作中应该尽量多使用快捷键，下面的这些快捷键是提高效率的好帮手，但很多书中都一带而过，甚至没有提及，请一定要牢牢记住下面的内容。

① Ctrl+J：复制当前图层到一个新层。

② J：切换到喷枪工具。

③ M：切换到选框工具。

④ []：在当前工具为画笔模式时（包括喷枪、画笔、铅笔、仿制图章历史画笔、橡皮及模糊和加深等工具），可依次增减笔头大小。

⑤ Shift+Backspace：调出填充对话框。

一开始，你可能无法记住所有的快捷键，可以使用 Photoshop 的工具提示来帮助你。方法是执行“编辑”→“预置”→“常规”命令，选择“显示工具提示”。这样，当鼠标移动到工具面板上时，工具名称及其快捷键就会出现，直到移走鼠标才会消失。

（2）无接缝贴图。无论是对 3D 图像或是网页的制作，无接缝贴图都是很重要的，都可以在 Photoshop 中轻易地完成。定制好图像后，选择“滤镜”→“其他”→“位移”命令，在水平和垂直方向上位移，一般设置位移量为图像大小的一半，最重要的是将未定义区域设为折回。在完成位移之后，用橡皮图章工具在图像的拼合处涂抹，消除接缝，然后将图像定义为图案。用这种图案填充方法就可以得到无缝的背景图像。

（3）为常用的组合命令定制动作。在处理图像的时候，很多情况下都会用到组合命令。例如，需要将一个多层图像的文档移动到另一个文档中。最快捷且安全的方法是：新建一个图层（Shift+Ctrl+N），拼合新建图层以下所有可见图层（Shift+Alt+Ctrl+E），选择全部图像（Ctrl+A），复制（Ctrl+C），取消选择（Ctrl+D），删除当前图层。这组命令是为图像建立一个映像，它包含了当前层之下的所有可见层图像，这时剪贴板上已经留下了图像的内容，只要把它粘贴在需要的文档中就可以了。这是个很典型的例子，尽管可以使用一系列的快捷键来完成这组命令，但还是推荐把它储存为一个动作，在动作选项面板中，可以为这个动作设置一个方便的功能键，如 Shift+F2，这样随时调用，非常方便，尤其是将图像从 Photoshop 导入到其他应用程序中时，可节约大量时间。Shift、Ctrl 和 F2～F12 这些键不同的组合可以定义很多动作，能够大幅提高效率。

（4）自动选择图层。这是相当不起眼的一个小动作，却非常有用，特别是在一些多图层的大型文件中，即使规规矩矩地为每个图层命名，按顺序叠放图层，在选择时也是较为麻烦的事情。这时，可以先选择移动工具，在工具选项中选择“自动选择图层”，这样，除了切片工具、路径组件选择工具、钢笔工具和抓手工具外，工具箱中的其他工具为当前所选时，按住 Ctrl 键时，当前工具都会暂时变成移动工具，单击画布上的任意对象，Photoshop 都会自动转到其所在图层，这样就可以进行操作了。在当前为移动工具时，即使不选择“自动选择图层”，只要按住 Ctrl 键，同样可以自由选择分属不同图层的图像内容。



（5）创建自定义笔刷。在 Photoshop 中有一项很有用但未被大多数人充分利用的资源，那就是创建自定义笔刷。新建一个文档，大小为要制作的笔刷大小，用黑色（也可以是彩色或不同的灰度，但这样制出的画笔颜色会较淡）绘制画笔图，如一朵小花、几颗小树等。然后将该图案选中，定义为新的画笔（如果没有选中，Photoshop 就会把画布上全部图案定义为画笔）。这样，就可以用这些自定义的画笔创作各种独特的图像了，特别是一些边框等装饰性的图案。

（6）镜头光晕效果。在初级的图像处理中，镜头光晕可说是最常用的修饰效果之一。一般情况下，可以直接在图像中使用“滤镜”→“渲染”→“镜头光晕”命令，来为图像增加气氛。那么，如果对某一次的镜头光晕效果非常满意，希望把它保存下来，可以利用图层混合的特性。先在图像上新建一层，用黑色填充，再执行镜头光晕滤镜，然后将这一层的图层混合模式设为“屏幕”，此时，黑色被隐去，得到单独的光晕效果图层。不过，这个方法的缺点是，不能在白色的图像上显示。

（7）光照效果滤镜。这个滤镜给人最大的惊喜在于它可以创作出各种各样逼真的纹理效果。其具体用法是：将一个简单的纹理放置到一个通道中，然后对某一图层应用“滤镜”→“渲染”→“光照效果”命令，在纹理通道中选择刚才存储纹理的通道，调整各种相应的光照设置，这样就能得到具有立体感的纹理效果。

（8）使用调整图层。谁也没有把握能够一次就将图像调整得十分完美，所以，除了养成良好的备份习惯之外，还应尽量使用调整图层。你可能会觉得麻烦，但这点麻烦和图像损坏的损失比较起来就微乎其微了。更重要的是，你可以在创作过程中随时做出新的调整而不必担心图像的损坏。

（9）快速复制技巧。在同一个文档中，确定当前为移动工具（或暂时为移动工具），按下 Alt 键的同时拖移对象，即可复制，按住 Shift 键可保证按 45° 的倍数移动；在不同的文档间，移动时按住 Shift 键，如果两个文档的大小相同，则对象可复制到新文档的相同的位置，如果文档大小不同，那么对象会复制到新文档的正中。用这种方法复制，不但方便，也可以减少剪贴板的使用，进一步节省系统资源。

（10）成果的积累。在使用 Photoshop 一段时间以后，一定会积累不少好的创意，如调整的渐变、等高线、样式等，还包括一些方便工作的色板等。这些只是暂时保存在 Photoshop 中，一旦重装，Photoshop 将会恢复到默认状态，这些自定义的东西会消失得无影无踪。这不但可惜，还会影响到你的工作（如定义好的色板丢失）。所以，最好每有得意之作时就用预设管理器保存下来，存放在专门的文件夹中。预设管理器可以保存画笔、色板、渐变、样式、图案、等高线、自定义形状，这样你的工作成果一样也不会丢失。重装之后，用相应的文件覆盖就可回到熟悉的工作环境中了。

综合实训四　微博随身之数据库基础

【问题情境】

本实训将介绍系统开发前的准备工作，主要包括数据库的设计、表的创建及数据源的



配置，同时还包括系统中非代码（如图片等）资源的准备工作。

【拓展知识】

1. 数据库设计

首先需要对整个系统的数据库进行设计，本系统包括 10 个表，分别为用户信息表（user）、好友关系表（friend）、日志表（diary）、相册表（album）、照片表（photo）、日志评论表（comment）、照片评论表（p_comment）、访客记录表（visit）、头像表（head）、最大编号表（max）等，各个表之间的关系如图 4-1 所示。

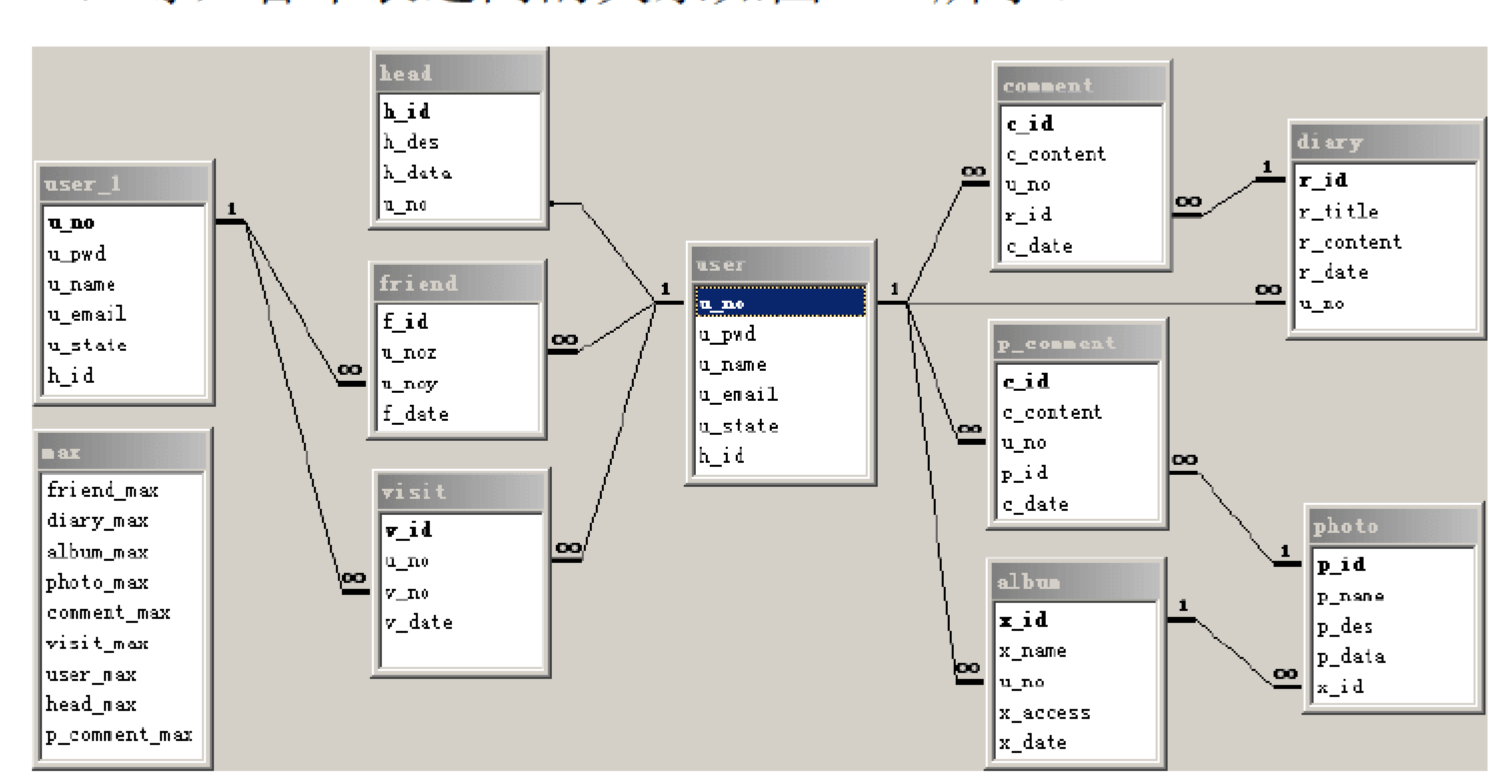


图 4-1　数据库结构

下面分别对每个表进行详细介绍。

（1）用户信息表：用于记录用户的相关信息，主要字段有用户 ID、用户密码、用户昵称、用户邮箱地址、用户状态、用户头像 ID。具体设计如表 4-3 所示。

表 4-3　用户信息表具体设计说明

字 段 名 称	数 据 类 型	字 段 大 小	是否为主键	是否可以为空	说　　明
u_no	int	N/A	是	否	用户 ID
u_pwd	varchar	16	否	否	用户密码
u_name	varchar	8	否	否	用户昵称
u_email	varchar	18	否	否	用户邮箱地址
u_state	text	N/A	否	否	用户状态
h_id	int	N/A	否	否	用户头像 ID

建立该表的 SQL 语句如下。

```
CREATE TABLE user(                                //用户表 user 的创建
        u_no          INT NOT NULL,               //用户 ID
        u_pwd         VARCHAR(16)  NOT NULL,      //用户密码
        u_name        VARCHAR(8),                 //用户昵称
        u_email       VARCHAR(18),                //邮箱地址
        u_state       TINYTEXT,                   //用户心情
        h_id          INT,                        //头像编号
        PRIMARY       KEY(u_no),                  //声明主键
```



```
        FOREIGN KEY(h_id) REFERENCES head(h_id)
);
```

（2）好友关系表：用于记录具有好友关系双方 ID 的表，主要包括的字段有用来唯一标识一组好友关系的编号、主人 ID、好友 ID 和日期。具体设计如表 4-4 所示。

表 4-4　好友关系表具体设计说明

字段名称	数据类型	字段大小	是否为主键	是否可以为空	说　　明
f_id	int	N/A	是	否	用来唯一标识一组好友关系的编号
u_noz	int	N/A	否	否	好友关系中的主人 ID
u_noy	int	N/A	否	否	好友关系中的好友 ID
f_date	timestamp	N/A	否	否	建立好友关系的日期

建立该表的 SQL 语句如下。

```
CREATE TABLE friend(                                  //好友关系表 friend 的创建
        f_id            INT NOT NULL,                 //好友关系编号
        u_noz           INT NOT NULL,                 //好友关系主人 ID
        u_noy           INT NOT NULL,                 //好友关系好友 ID
        f_date          TIMESTAMP,                    //好友关系建立日期
        PRIMARY KEY(f_id),
        FOREIGN KEY(u_noz) REFERENCES user(u_no),
        FOREIGN KEY(u_noy) REFERENCES user(u_no)10    //声明外键
);
```

（3）日志表：用于记录用户发表的日志，主要包括的字段有唯一标识日志的编号、日志标题、日志内容、日志发布的日期、日志所属用户的 ID。具体设计如表 4-5 所示。

表 4-5　日志表具体设计说明

字段名称	数据类型	字段大小	是否为主键	是否可以为空	说　　明
r_id	int	N/A	是	否	唯一标识日志的编号
r_title	varchar	18	否	否	日志标题
r_content	text	N/A	否	否	日志内容
r_date	timestamp	N/A	否	否	日志发布的日期
u_no	int	N/A	否	否	日志所属用户的 ID

建立该表的 SQL 语句如下。

```
CREATE TABLE diary(                            //日志表 diary 的创建
        r_id            INT  NOT NULL,         //日志编号
        r_title         VARCHAR(18) NOT NULL,  //日志标题
        r_content       TEXT NOT NULL,         //日志内容
        r_date          TIMESTAMP,             //发布日期
        u_no            INT NOT NULL,          //所属用户 ID
        PRIMARY         KEY(r_id),
        FOREIGN         KEY(u_no) REFERENCES user(u_no)
);
```

（4）相册表：用于记录用户相册的相关信息，主要包括的字段有相册编号、相册名称、相册所属用户的 ID、相册访问权限、相册创建日期。具体设计如表 4-6 所示。

表 4-6　相册表具体设计说明

字段名称	数据类型	字段大小	是否为主键	是否可以为空	说　明
x_id	int	N/A	是	否	相册编号
x_name	varchar	18	否	否	相册名称
u_no	int	N/A	否	否	相册所属用户的 ID
x_access	int	N/A	否	是	相册访问权限（0 表示公开，1 表示好友可见，2 表示仅用户可见，默认为 0）
x_date	timestamp	N/A	否	否	相册创建日期

建立该表的 SQL 语句如下。

```
CREATE TABLE album(                              //相册表 album 的创建
    x_id        INT NOT NULL,                    //相册编号
    x_name      VARCHAR(18) NOT NULL,            //相册名称
    u_no        INT NOT NULL,                    //所属用户 ID
    x_access    INT DEFAULT 0,                   //0：公开，1：好友，2：仅自己可见
    x_date      TIMESTAMP,                       //创建日期
    PRIMARY KEY(x_id),
    FOREIGN KEY(u_no) REFERENCES user(u_no)
);
```

（5）照片表：用于记录相册中的照片信息，主要包括的字段有照片编号、照片名称、照片描述、照片的二进制数据、照片所属相册。具体设计如表 4-7 所示。

表 4-7　照片表具体设计说明

字段名称	数据类型	字段大小	是否为主键	是否可以为空	说　明
p_id	int	N/A	是	否	照片编号
p_name	varchar	18	否	否	照片名称
p_des	varchar	50	否	否	照片描述
p_data	mediumblob	N/A	否	否	照片的二进制数据
x_id	int	N/A	否	否	照片所属相册

建立该表的 SQL 语句如下。

```
CREATE TABLE photo(                              //照片表 photo 的创建
    p_id        INT NOT NULL,                    //照片编号
    p_name      VARCHAR(18) NOT NULL,            //照片名称
    p_des       VARCHAR(50) NOT NULL,            //照片描述
    p_data      MEDIUMBLOB,                      //照片二进制数据
    x_id        INT NOT NULL,                    //所属相册
    PRIMARY KEY(p_id),
    FOREIGN KEY(x_id) REFERENCES album(x_id)
);
```



（6）日志评论表：用于记录用户对日志的评论，主要包括的字段有评论编号、评论内容、评论者 ID、评论所属的日志 ID、评论日期。具体设计如表 4-8 所示。

表 4-8　日志评论表具体设计说明

字段名称	数据类型	字段大小	是否为主键	是否可以为空	说　　明
c_id	int	N/A	是	否	评论编号
c_content	text	N/A	否	否	评论内容
u_no	int	N/A	否	否	评论者 ID
r_id	int	N/A	否	否	评论所属的日志 ID
c_date	timestamp	N/A	否	否	评论日期

建立该表的 SQL 语句如下。

```
CREATE TABLE comment(                          //日志评论表 comment 的创建
        c_id          INT NOT NULL,            //日志评论编号
        c_content     TEXT NOT NULL,           //评论内容
        u_no          INT NOT NULL,            //评论者 ID
        r_id          INT NOT NULL,            //日志 ID
        c_date        TIMESTAMP,               //评论日期
        PRIMARY       KEY(c_id),
        FOREIGN       KEY(u_no) REFERENCES user(u_no),
        FOREIGN       KEY(r_id) REFERENCES diary(r_id)
);
```

（7）照片评论表：用于记录用户对照片的评论，主要包括的字段有评论编号、评论内容、评论者 ID、评论所属的照片编号、评论日期。具体设计如表 4-9 所示。

表 4-9　照片评论表具体设计说明

字段名称	数据类型	字段大小	是否为主键	是否可以为空	说　　明
c_id	int	N/A	是	否	评论编号
c_content	text	N/A	否	否	评论内容
u_no	int	N/A	否	否	评论者 ID
r_id	int	N/A	否	否	评论所属的照片编号
c_date	timestamp	N/A	否	否	评论日期

建立该表的 SQL 语句如下。

```
CREATE TABLE p_comment(                   //照片评论表 p_comment 的创建
        c_id          INT NOT NULL,       //评论编号
        c_content     TEXT NOT NULL,      //评论内容
        u_no          INT NOT NULL,       //评论者 ID
        p_id          INT NOT NULL,       //评论所属照片编号
        c_date        TIMESTAMP,          //评论日期
        PRIMARY KEY(c_id),                //设置主键
        FOREIGN KEY(u_no) REFERENCES user(u_no),
```

```
        FOREIGN KEY(p_id) REFERENCES photo(p_id)
);
```

（8）访客记录表：用于记录每个用户被其他用户访问的情况，主要包括的字段有访问记录编号、被访问者 ID、访问者 ID 和访问日期。具体设计如表 4-10 所示。

表 4-10　访客记录表具体设计说明

字段名称	数据类型	字段大小	是否为主键	是否可以为空	说　明
v_id	int	N/A	是	否	访问记录编号
u_no	int	N/A	否	否	被访问者 ID
v_no	int	N/A	否	否	访问者 ID
v_date	timestamp	N/A	否	否	访问日期

建立该表的 SQL 语句如下。

```
CREATE TABLE visit(                                   //访客记录表 visit 的创建
        v_id        INT NOT NULL,                     //访问记录编号
        u_no        INT NOT NULL,                     //被访问者 ID
        v_no        INT NOT NULL,                     //访问者 ID
        v_date      TIMESTAMP,                        //访问日期
        PRIMARY KEY(v_id),                            //声明主键
        FOREIGN KEY(u_no) REFERENCES user(u_no),      //声明外键
        FOREIGN KEY(v_no) REFERENCES user(u_no)       //声明外键
);
```

（9）头像表：用于记录用户可用的头像相关信息，主要包括的字段有头像编号、头像描述、头像图片的二进制数据、头像上传者 ID。具体设计如表 4-11 所示。

表 4-11　头像表具体设计说明

字段名称	数据类型	字段大小	是否为主键	是否可以为空	说　明
h_id	int	N/A	是	否	头像编号
h_des	varchar	40	否	否	头像描述
h_data	mediumblob	N/A	否	否	头像图片的二进制数据
u_no	int	N/A	否	否	头像上传者 ID

建立该表的 SQL 语句如下。

```
CREATE TABLE head(                                 //头像表 head 的创建
        h_id        INT NOT NULL,                  //头像编号
        h_des       VARCHAR(40) NOT NULL,          //头像描述
        h_data      MEDIUMBLOB NOT NULL,           //头像图片的二进制数据
        u_no        INT,                           //头像上传者 ID
        PRIMARY KEY(h_id),                         //声明主键
        FOREIGN KEY(u_no) REFERENCES user(u_no)    //声明外键
);
```

（10）最大编号表：用于记录各个表中主键编号的当前最大值，当需要向这些表中插

入新数据时，首先取得该表中对应字段的值，将其加 1 并作为新插入记录的编号。具体设计如表 4-12 所示。

表 4-12　最大编号表具体设计说明

字 段 名 称	数据类型	字段大小	是否为主键	是否可以为空	说　　明
friend_max	int	N/A	否	是	好友表的最大编号。默认为 0
diary_max	int	N/A	否	是	日志表的最大编号。默认为 0
album_max	int	N/A	否	是	相册表的最大编号。默认为 0
photo_max	int	N/A	否	是	照片表的最大编号。默认为 0
comment_max	int	N/A	否	是	日志评论表的最大编号。默认为 0
p_comment_max	int	N/A	否	是	照片评论表的最大编号。默认为 0
visit_max	int	N/A	否	是	访问记录表的最大编号。默认为 0
user_max	int	N/A	否	是	用户表的最大编号。默认为 0
head_max	int	N/A	否	是	头像表的最大编号。默认为 0

建立该表的 SQL 语句如下。

```
CREATE TABLE max(                                          //最大编号表 max 的创建
    friend_max        INT NOT NULL DEFAULT 0,   //存放好友表的最大编号
    diary_max         INT NOT NULL DEFAULT 0,   //存放日志表的最大编号
    album_max         INT NOT NULL DEFAULT 0,   //存放相册表的最大编号
    photo_max         INT NOT NULL DEFAULT 0,   //存放照片表的最大编号
    comment_max       INT NOT NULL DEFAULT 0,   //存放日志评论表的最大编号
    visit_max         INT NOT NULL DEFAULT 0,   //存放访问记录表的最大编号
    user_max          INT NOT NULL DEFAULT 0,   //存放用户表的最大编号
    head_max          INT NOT NULL DEFAULT 0,   //存放头像表的最大编号
    p_comment_max     INT NOT NULL DEFAULT 0    //存放照片评论表的最大编号
);
```

2. 表的创建和测试数据的插入

在前面已经介绍了数据库中表的设计，下面将介绍如何在 MySQL 数据库中创建这些表并插入原始数据，其步骤如下。

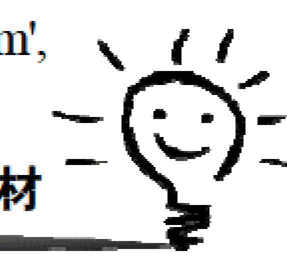
（1）启动 MySQL 服务，打开 MySQL 客户端 MySQL Command Line Client。如果在 MySQL 安装时设置了密码，则输入设定好的密码进入 MySQL 的命令行。

（2）在命令行输入“create database kdwb;”创建一个新的数据库，输入“use kdwb”进入新建的数据库中。

（3）依次执行前面创建各个表的语句，完成系统数据表的创建。

（4）数据库创建完成之后，向数据库中插入一些测试数据。测试数据的插入代码如下。

```
/*向 user 表插入数据*/
INSERT  INTO  user(u_no,u_pwd,u_name,u_email,u_state)  VALUES('2008','123',' 王 强 ','www@126.com','I am 快乐.');
INSERT INTO user(u_no,u_pwd,u_name,u_email,u_state) VALUES('2009','123','Tom','www@126.com',
```

```
'I am sad.');
INSERT INTO user(u_no,u_pwd,u_name,u_email,u_state) VALUES('2010','123','Jerry','www@126.com','I am busy.');
/*向 friend 表中插入数据*/
INSERT INTO friend(u_noz,u_noy) VALUES('2008','2009');
INSERT INTO friend(u_noz,u_noy) VALUES('2008','2010');
INSERT INTO friend(u_noz,u_noy) VALUES('2008','2029');
INSERT INTO friend(u_noz,u_noy) VALUES('2008','2030');
/*向 diary 表中插入数据*/
INSERT INTO diary(r_title,r_content,u_no) VALUES('工作日志 1','今天天气不好，阴沉沉的。','2008');
INSERT INTO diary(r_title,r_content,u_no) VALUES('工作日志 2','今天好冷。','2008');
INSERT INTO diary(r_title,r_content,u_no) VALUES('旅游日志','今天去看大海了。','2009');
/*向 album 表中插入数据*/
INSERT INTO album(x_name,u_no) VALUES('正定游','2008');
INSERT INTO album(x_name,u_no) VALUES('我的家人','2009');
/*向 comment 表里插入数据*/
INSERT INTO comment(c_content,u_no,r_id) VALUES('你写的日志太好了，我很稀饭哦','2009',SELECT r_id FROM diary WHERE u_no='2008');
/*向 visit 表插入数据*/
INSERT INTO visit(u_no,r_id) VALUES('2008');
/*向 max 表中插入数据*/
INSERT INTO max VALUES(0,0,0,0,0,0,2020); /*向最大编号表中插入唯一的记录*/
```

3. 数据源的配置

本系统使用 Tomcat 作为 Web 应用服务器，并使用数据源连接池进行数据库的连接，因此需要配置数据源，其步骤如下。

（1）添加数据库驱动，在 Tomcat 安装目录下的 lib 目录中复制 MySQL 数据库驱动中的 JAR 包——mysql-connector-java-5.1.12-bin.jar。

（2）打开 Tomcat 安装目录下的 conf 目录，修改该目录下的 server.xml 文件，在该文件的</host>标记之前插入如下代码。

```
<Context path="/KDWB" docBase="D:/MyWork/KDWB"
dubug="5" reloadable="true" crossContext="true" workDir=""><!--声明上下文-->
        <Resource name="jdbc/KDWB" auth="Container" type="javax.sql.DataSource"
        maxActive="100" maxIdle="30" maxWait="10000" username="" password=""
        driverClassName="org.gjt.mm.mysql.Driver"
        url="jdbc:mysql://localhost/kdwb"/>                    <!--声明数据源-->
</Context>
```

（3）打开 Web 端的项目文件夹下的 WEB-INF 目录中的 web.xml，在其中添加如下配置代码。



```
<resource-ref>                                                    <!--声明资源引用-->
    <description>DB Connection</description>                      <!--资源描述-->
    <res-ref-name>jdbc/mysql</res-ref-name>                       <!--数据源名称-->
    <res-type>javax.sql.DataSource<res-type>                      <!--资源类型-->
    <res-auth>Container</res-auth>
</resource-ref>
```

📖 小知识四：数据库标准化与范式

标准化是 IT 数据库专业人士的戒律之一，数据建模工程师、数据库管理员和 SQL 开发者都必须遵守这一戒律。我们应尽早就了解它的原理和范式。

对大部分数据库进行研究发现：大多数数据库至多执行了第三范式（3NF），很少有数据库执行了更高范式，如 Boyce-Codd 范式（BCNF）、第四范式（4NF）和第五范式（5NF）。为什么大多数数据库设计员没有超出 3NF 呢？

（1）范式简介

为了回答上述问题，了解 3NF、BCNF、4NF 和 5NF 之间的区别很重要。以下为每个范式的准确定义。

- ☑ **第一范式（1NF）**，1NF 要求每个表必须有一个首要键，即最少的一组属性，它与每条记录一一对应。通过适当定义键属性和非键属性，删除重复的组（不同记录似乎需要不同次重复的数据种类）。注：每个属性必须包含单独一个值，而非一组值。
- ☑ **第二范式（2NF）**，数据库必须满足 1NF 的所有要求。另外，如果一个表有一个复合键，所有属性必须与整个键相关联。而且，在表的多行之间多余重复的数据会被移动到一个单独的表中。
- ☑ **第三范式（3NF）**，数据库必须满足 2NF 的所有要求。另外，存储在表中的数据不得依赖表的任何域，必须唯一依赖于首要键。而既依赖首要键，又依赖其他域的数据会被移动到一个单独的表中。
- ☑ **Boyce-Codd 范式（BCNF）**，除对一个候选键扩展集（称作一个超级键）存在属性函数依赖外，不存在其他非平凡函数依赖。
- ☑ **第四范式（4NF）**，除对一个候选键扩展集存在属性组函数依赖外，不存在其他非平凡多值函数依赖。如果有且只有一个表符合 BCNF，同时多值依赖为函数依赖，此表才符合第四范式。4NF 删除了不必要的数据结构——多值依赖。
- ☑ **第五范式（5NF）**，不得存在不遵循键约束的非平凡连接依赖。如果有且只有一个表符合 4NF，同时其中的每个连接依赖被候选键所包含，此表才符合 5NF。

（2）单值与多值依赖

下面详细介绍这两种类型依赖的不同。

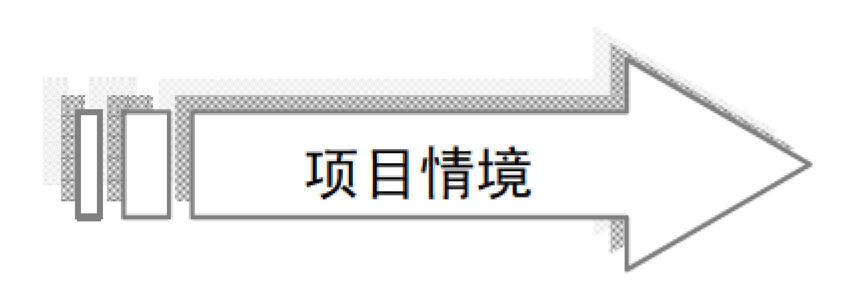
例如，一名仅在组织的一个部门工作的员工就属于单值依赖。这名员工可以在部门之间调动，但不能同时为两个部门工作。

在与每个地址有关的数据库中，都可以发现多值依赖的例子。通常情况下，在 Programmers 表中有 City、State 和 Country 这些列。这些地址可能为文本，或者在方便查找

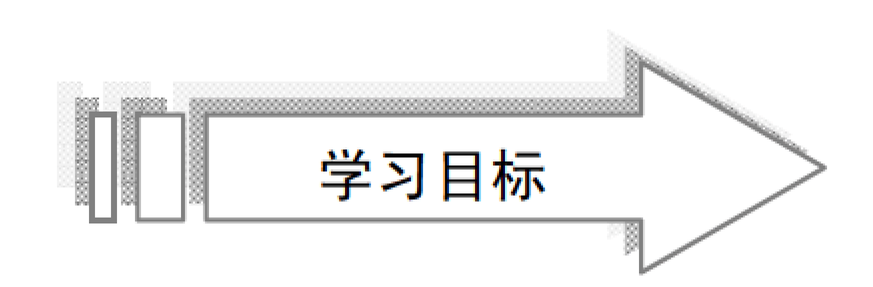
的情况下，也可能为整数值。City 查找城市表、State 查找州表、Country 查找国家表。这种安排会带来无用地址的风险问题，如芝加哥、纽约、加拿大。这是因为这里的依赖是多值依赖。

完全标准化的版本可能会把 State 列移动到城市表，把 Country 列移动到国家表，在 Programmers 表中只留下 City 列。我们可以建立一个连接三个表的查询，并提前执行这个查询，这样用户就可以根据需要选择合适的城市。

项目五
游戏的整体架构



本项目将对游戏的架构进行简单介绍，因本游戏涉及的类较多，且这些类有些具有相同的特征或实现相同的功能，而有些则协作服务于同一个模块。然而设计好游戏的架构可以使开发的过程更加规范，少走弯路。



- 熟悉游戏的功能模块和整体架构
- 了解并掌握实现游戏功能的各个类

任务 9 游戏的模块架构

【任务情境】

在本任务中将会对游戏的功能模块的结构做一个简单介绍。

【相关知识】

本游戏中各模块的结构如图 5-1 所示。

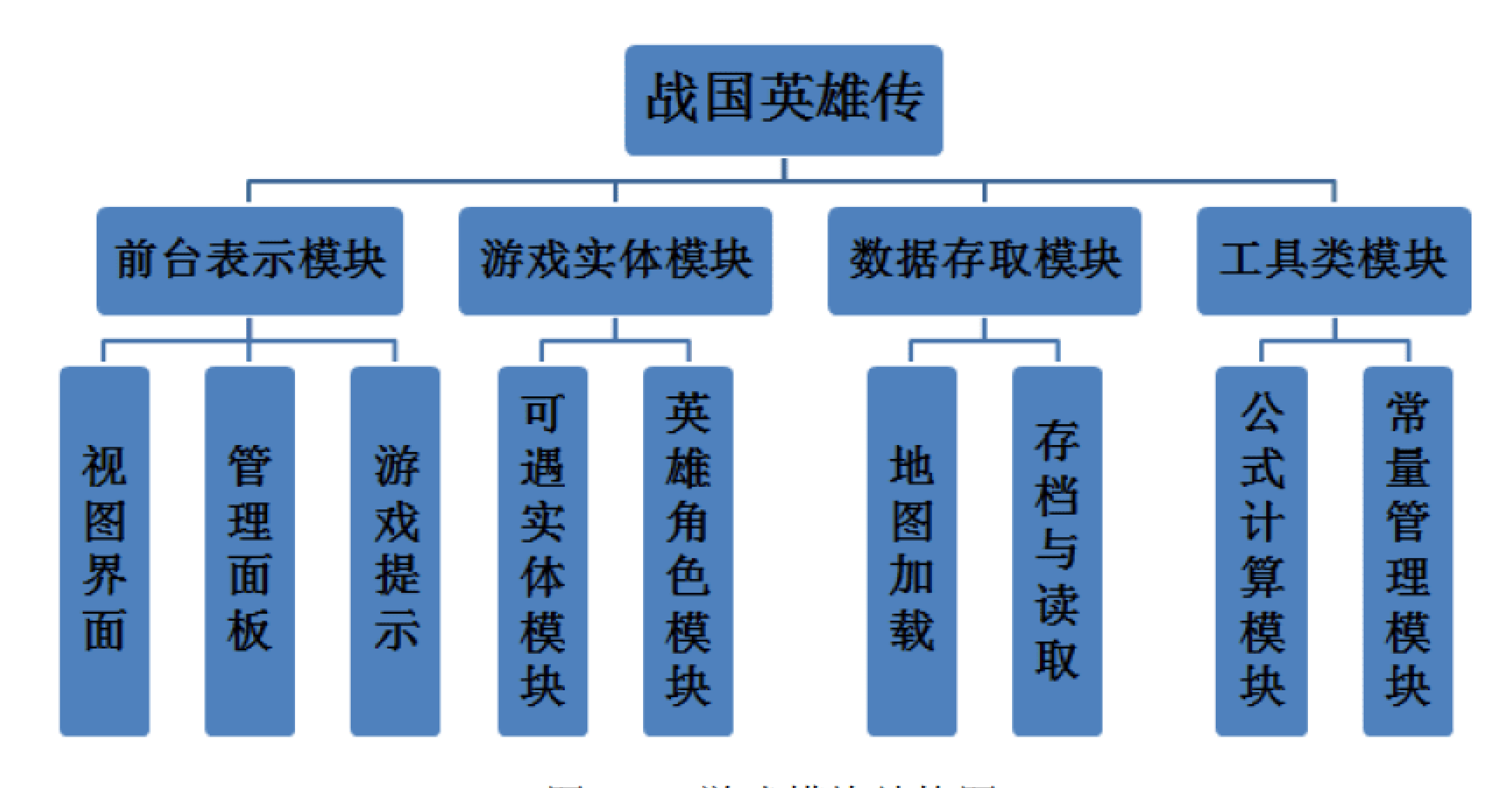


图 5-1 游戏模块结构图

- ☑ 前台表示模块主要用于游戏界面的渲染，包括视图界面、管理面板和游戏提示 3 个模块。视图界面主要为游戏中出现的界面，包括加载界面、菜单等视图；管理面板主要是向玩家提供查看和修改游戏数据的面板，包括将领管理、城池管理等面板；游戏提示负责在游戏进行中向玩家显示粮草危机、游戏失败等信息。
- ☑ 游戏实体模块主要用于后台游戏逻辑，包括英雄角色模块和可遇实体模块。英雄角色模块主要负责控制英雄的移动以及检测是否与地图中可遇实体发生碰撞。英雄角色模块还包括英雄技能模块，技能模块包括普通技能（如伐木、打鱼等）和计谋（如随心步、回头是岸等）；可遇实体模块是地图上那些可以被英雄遇到并激发一系列动作的实体（如遇到稻田可以收获、遇到城池可以攻打等）。
- ☑ 数据存取模块包括地图加载模块和存档与读取模块。地图加载主要是将从地图设计器获得的地图信息文件加载到游戏中并生成地图矩阵的过程；存档与读取模块用于将游戏的状态保存到文件并在需要的时候读取到游戏中。
- ☑ 工具类模块将自身的静态成员或方法提供给游戏中的其他类使用。本游戏中的工具类包括对常量进行统一管理的 ConstantUtil 类和用于存储游戏公式的 GameFormula 类，这里的游戏公式主要包括战斗的输赢、英雄体力增加等计算。



任务 10　游戏各个类的简要介绍

【任务情境】

前面的任务对游戏的总体架构进行了介绍，在了解了游戏中的功能模块之后，本任务将对实现这些功能模块的具体类进行简单说明。

【相关知识】

1. 前台表示模块的类结构

前台表示模块的类结构如图 5-2 所示。

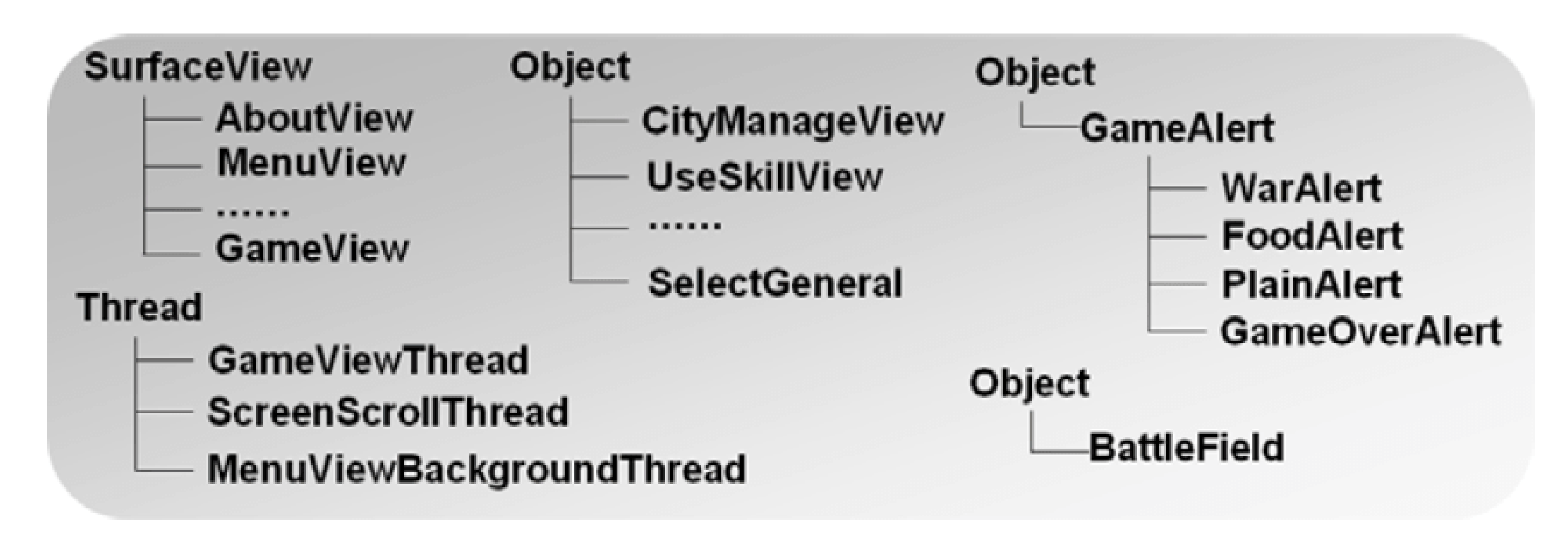


图 5-2　前台表示模块类结构图

- ☑ AboutView、MenuView 和 GameView 等一系列继承自 SurfaceView 的视图是组成游戏视图界面模块的主要部分，这些视图分别用来显示不同的内容，如菜单、帮助、滚动卷轴等。继承自 Thread 的 GameViewThread、ScreenScrollThread 和 MenuViewBackgroundThread 类负责修改上述视图的后台数据。BattleField 类的功能是在 GameView 界面绘制战斗动画。
- ☑ 游戏的管理面板模块中包括的类有 CityManageView、UseSkillView 和 SelectGeneral 等继承自 Object 的自定义类，这些类功能是在 GameView 中绘制不同的管理面板，如个人属性面板、将领面板、城池面板等。
- ☑ 游戏提示模块是由 GameAlert 的子类组成的，GameAlert 是继承自 Object 的自定义类，其子类的主要功能是在 GameView 显示游戏的提示信息，如有敌人偷袭城池、粮草发生危机或游戏结束时分别可以通过 WarAlert、FoodAlert 和 GameOverAlert 提示玩家进行相应操作。当只需要向玩家提示信息而不需要进行其他后续操作时，可以通过 PlainAlert 来实现。

2. 游戏实体模块的类结构

通过任务 9 可以了解到，游戏的实体模块部分包括地图的可遇实体模块和英雄角色模块，这两个子模块的类结构如图 5-3 所示。

☑ 本游戏地图中要绘制的图元也是 MyDrawable 对象，且本游戏中有一种特殊的 MyDrawable 对象可以与英雄相遇，在相遇之后可以触发特定的操作。这种可遇的实体对象均继承自 MyMeetableDrawable 类，该类为继承自 MyDrawable 类的抽象类。游戏将会为地图中所有的可遇实体创建一个相应的 MyMeetableDrawable 对象，例如，英雄在地图上遇到森林时，ForestDrawable 对象会调用相关的方法来与玩家进行交互。

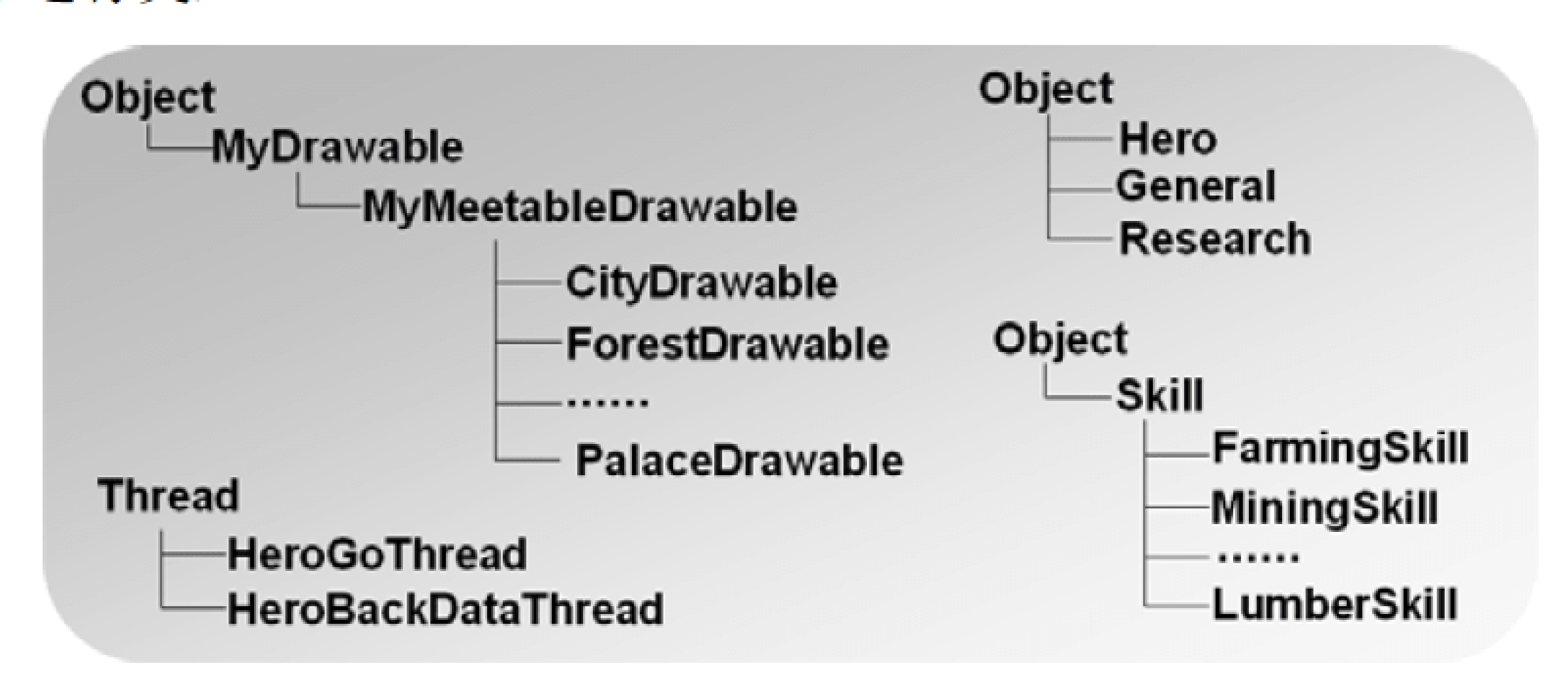


图 5-3　子模块类结构图

☑ 游戏中另外一个重要的实体对象就是 Hero 对象，Hero 对象中封装了英雄的属性，如所有城池、手下将领、金钱粮草等。General 和 Research 类均为 Hero 类的辅助类，General 类为将领类，其中封装了防御力、武力和忠诚度等信息。Research 为科研类，当英雄建造箭垛或战车时，就会创建一个 Research 类来负责在后台进行建造。

☑ 继承自 Thread 类的 HeroGoThread 和 HeroBackDataThread 为 Hero 的附属线程，HeroGoThread 负责根据掷出的骰子点数将英雄移动指定的距离，同时进行可遇检查。HeroBackDataThread 负责修改个人数据，如定时消耗粮食、增加科研项目的进度以及随机产生攻击英雄城池事件，该线程的休眠时间很长，因为这些工作并不需要频繁地执行。

☑ 在游戏中，英雄具有一定的技能，这些技能均为继承自抽象类 Skill 的子类对象，Skill 类为继承自 Object 的自定义类，其中包含一些需要子类实现的抽象方法。在 Hero 类中会维护一个 Skill 对象列表来存储英雄学会的技能。

3. 数据存取模块的类结构

数据存取模块涉及到的类及其结构如图 5-4 所示。

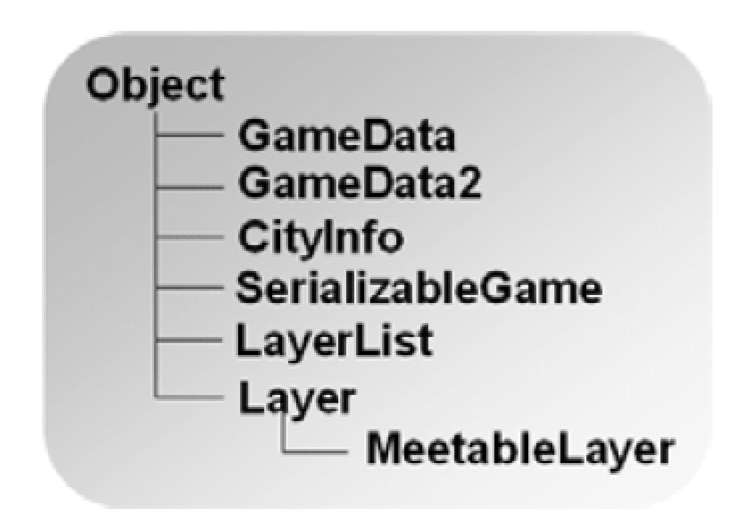


图 5-4　数据存取模块类结构图



☑ GameData 和 GameData2 类实现的功能类似，都是加载存放在 Assert 中的地图信息文件，根据其内容创建游戏地图的 MyDrawable 二维数组。二者不同的是 GameData 负责加载并生成下层平铺层的地图，而 GameData2 类负责加载并生成上层平铺层的地图，地图中的可遇实体对象位于上层平铺层。

☑ Layer 类为继承自 Object 的自定义类，该类用于维护一个图层的绘制矩阵（MyDrawable 矩阵）和不可通过矩阵。MeetableLayer 类继承自 Layer 类，除了进行和 Layer 相同的工作外，还负责维护该图层的可遇矩阵，英雄移动到某个位置后通过该可遇矩阵来判断是否与地图中的可遇实体发生相遇。

☑ SerializableGame 类的主要功能是存储游戏状态和加载已存储的游戏，该类提供了用于实现这些功能的静态方法以及检测游戏是否存过档的方法。

☑ CityInfo 用于存储城池信息，其中封装了诸如城池名称、人口、所属国、守城将领、兵力等信息，游戏中创建 CityDrawable 对象时会从 CityInfo 中读取信息。

4. 游戏工具类的类结构

游戏工具类模块比较简单，其结构如图 5-5 所示，主要涉及到的类为 ConstantUtil 和 GameFormula。ConstantUtil 类的功能在前面已经提及过，主要是对游戏中用到的常量进行统一管理。GameFormula 类中提供了一系列的静态方法用于实现诸如计算城池防御力和攻击力、按一定算法增加英雄体力等功能。

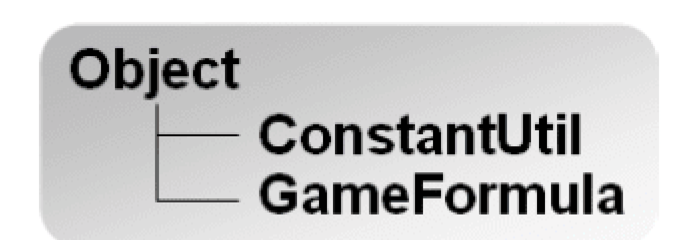


图 5-5　游戏工具类的类结构图

【项目小结】

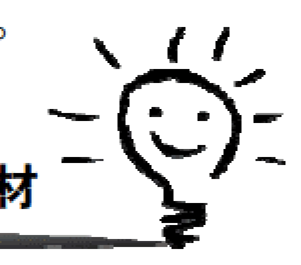
本项目主要介绍了“战国英雄传”游戏的整体架构，包括其模块架构和各个类的简要介绍。不同的游戏根据其复杂度不同会有不同的架构和类，这些都要在正式进入开发之前设计好，如共需要几类来实现、分别是哪几类、各类有什么特征、实现什么功能等。好的游戏架构可以使开发过程更规范、更高效。

📖 小知识五：整体把握游戏构建模块，创造富有意义的体验

通常，我们在设计游戏时的很多决策都是无意识或凭直觉进行，这有时会正中要点，也许会导致体验缺乏平衡性，受突发奇想左右或缺乏连贯性。把握设计决策的整体运作方式是否能够促使我们设计出更富意义的游戏体验？假设我们将游戏视作沟通媒介，而沟通就需要清晰且具有连贯性。

我们在设计游戏过程中所做的决策不要局限于它们是否具有“趣味性”。这听起来平淡无奇，但设计决策背后的逻辑常被忽略，鲜少被认真看待。在音乐中，我们能够凭直觉知晓小调代表“悲伤”，大调传递“高兴”，表达欢乐或充满希望的情感。

和音乐一样，游戏也存在系列传达理念和情感的潜在规则，其中有些源自其他媒介。



辨别、学习和操作这些规则能够促使我们设计出更富意义的体验。幸运的是，如今我们能够通过若干工具发现和应用这些规则。在学术领域，符号学是学习意义系统的学科。符号学者可以选择学习游戏中的意义系统。下面主要涉及如何基于符号学原理进行游戏设计，同时还包括游戏通过符号和风格原理创造富有意义的连贯体验的若干方式。

（1）色调和风格

就游戏呈现的元素来看，我们也许会觉得："这看起来很棒！"当然也许这款游戏看起来真的很不错。但图像和声音决策远比"呈现外观"的美学理论更深入。例如，追溯到所有游戏都是 2D 模式的时代。很多玩家可能都还记得这些 2D 游戏，他们会将这些游戏的所有知识都带入当前的游戏中。所以假设 2011 年你决定制作采用 2D 而非 3D 模式的游戏，游戏不仅会看起来"很棒"，而且会唤起玩家关于这些传统游戏的记忆和情感。这是否就是我们期望从作品中得到的？我们是否希望玩家将我们的游戏同传统作品相比较？我们是否希望他们记住初次体验这些游戏的感觉？或者我们只是想要搭载更适合 2D 内容的平台？游戏设计和我们所做出的美学决策具有完整的"意义"，这有时全凭直觉——慢慢意识到这些意义也可以是强大工具。

（2）规则和机制

游戏和传统媒介的一大区别在于支配玩家游戏互动的规则。这些规则支配游戏空间，赋予操作相应含义。游戏中，你能够彻底改变物理学原理，支配玩家生死的情境，设计整个经济体系，呈现前所未有的体验。值得注意的是，所有围绕游戏机制的决策都具有相应含义，它们同其他游戏元素相互配合（游戏帮助：包括声音、画面和输入内容），期望以微妙方式创造游戏体验。设计规则和机制主要基于内容是否有趣，但其中的各元素也会赋予游戏一定意义。例如，需玩家依靠策略手段取胜的游戏会创造不同的"世界观"，区别于通过武力获胜的模式。换而言之，这不仅是个体验规则，还能通过呈现特定世界观赋予内容一定意义。

（3）整体配合

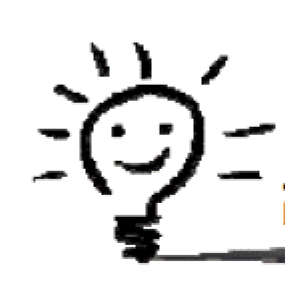
从所有这些设计内容中，我们能够逐渐看到各要素间存在的微妙联系，以及它们如何相互配合创造游戏世界。回到小调和大调的例子，我们会逐渐发现游戏存在的相似规则。和其他媒介一样，创建游戏意味着需要进行众多艰难抉择。我们应该深入挖掘这些决策的影响，清楚保留及去除内容会如何影响游戏功能及体验"意义"。

这里我想要表达的是，通过将游戏分解成若干构建模块，把握它们的运作方式，我们能够系统地进行游戏设计，这样游戏元素就能够相互配合，向玩家呈现连贯、有意义且富有趣味的游戏体验。

综合实训五　微博随身之整体架构

【问题情境】

在前面已经介绍了微博随身系统背景及开发前的数据库设计等准备工作，本实训将对



微博随身 Web 端的功能进行初步的预览，并对 Web 端的总体架构进行简单的介绍。

【拓展知识】

1. Web 端系统功能

下面将对微博随身的 Web 端功能进行预览，Web 端的主要功能如下。

（1）打开主页后，用户可以在右上角的模块中选择登录或注册新用户的功能。

（2）在登录界面输入用户名和密码并成功登录后，页面中会显示用户的昵称和心情等基本信息，同时还会提供用户好友列表和最近访客的显示，用户可以通过标签浏览这些信息。

（3）用户登录后可以发布新的日志和更新自己的心情，通过单击“查看日志”超链接，可以查看自己发布过的日志及好友们对日志的评论，同时还可以管理自己的日志，将日志删除或进行编辑。

（4）用户登录后单击“查看相册”超链接可以查看自己的相册，同时还可以创建新的相册、上传图片及管理相册的权限。相册的权限包括完全公开、好友可见及仅主人可见三种。

用户在相册查看页面可以删除指定的图片，还可以修改当前相册的访问权限，具有特定访问权限的相册将根据来访者的身份决定是否向访问者显示该相册。

（5）用户通过单击“上传图片”超链接进入图片上传页面，在图片上传页面中用户可以选择上传的相册或者单击“新建相册”创建一个新相册。选择了一个相册并填好图片的名称等信息后，单击“浏览”按钮可从电脑中选择要上传的图片，单击“确定”按钮即可开始上传。图片上传成功后，用户可以选择继续上传或是返回相册查看图片。

（6）用户除了管理自己的日志和相册外，还可以修改包括头像在内的个人资料。单击“修改资料”超链接可以打开个人资料修改和更改头像页面，如果用户对当前提供的可换头像不满意，可以自己上传喜欢的头像并将其设置为自己的头像。

（7）用户登录后可以按照昵称搜索特定的用户，搜索结果将显示在屏幕的中央。

在搜索结果中将列出符合搜索条件的用户的昵称、头像和心情等。如果搜索到的微博用户不是自己的好友，则会显示“添加好友”超链接，通过单击该超链接可以添加特定用户为自己的好友。

（8）用户登录后通过单击“我的好友”和“最新访客”标签可以查看用户的好友和最近访问者，单击这些联系人的昵称可以访问其微博，访问页面同个人的日志相册管理页面类似，只是不会提供修改、删除等只对主人开放的功能。

2. 系统目录结构

前面已经对微博随身 Web 端的主要功能进行了简单的预览。在进行系统的具体开发之前，还需要介绍系统的目录结构，目录结构可以很好地说明系统的开发原理。本系统 Web 端的目录组织结构如图 5-6 所示。

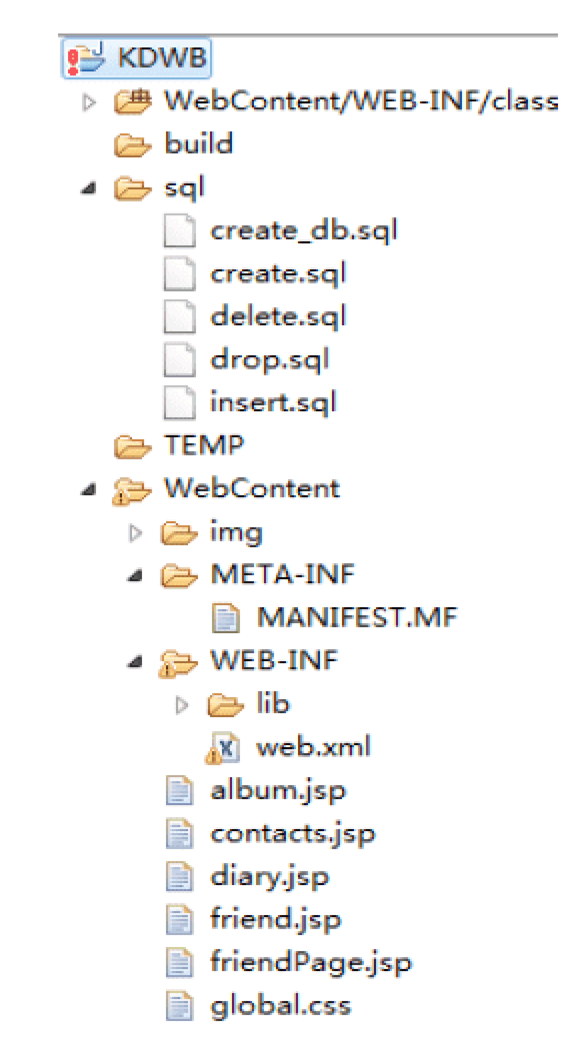


图 5-6　Web 端目录结构图



3. 系统总架构

在前面已经介绍了微博随身 Web 端的目录组织结构，下面将对 Web 端系统的功能结构进行介绍，Web 端的系统功能框架如图 5-6 所示。从该图中可以看出系统中各个 JSP 文件和 Servlet 等.class 文件之间的关系。

在如图 5-6 所示中，MyServlet 和 FileUploadServlet 为控制器，负责处理浏览器客户端发来的各种请求，DBUtil 为负责处理具体业务的工具类，各种 JSP 文件负责前台显示。图中的各种动作均用字母来表示，具体含义如表 5-1 所示。

表 5-1　动作编号内容说明

动作编号	动作内容	动作编号	动作内容	动作编号	动作内容
a	用户登录	i	创建相册	q	请求日志列表
b	转到注册页面	j	修改相册权限	r	编辑日志
c	好友列表	k	获得指定相册相片	s	删除日志
d	访客列表	l	删除相片	t	评论日志
e	提交注册信息	m	评论相片	u	显示日志
f	显示用户资料	n	显示相册列表	v	显示日志评论
g	修改个人资料	o	显示相片列表	w	上传相片
h	获得相册列表	p	显示相片评论	x	上传头像


提示

由于本系统中的动作很多，无法在表 5-1 中一一列举，仅选用了一些具有代表性的动作。

4. Android 端功能预览

下面开始将对微博随身 Android 端各个功能模块的开发进行介绍。首先对 Android 端的功能进行预览，接下来对 Android 端的总体架构进行介绍。

以下对微博随身的 Android 端的功能进行演示。Android 端的主要功能如下。

（1）应用程序启动后，首先显示的是登录界面，在登录界面中输入账号和密码，如果希望程序记住自己的账号和密码，可以选中“记住我”复选框进行设置。填写好账号和密码后，单击“登录”按钮连接服务器进行验证。

（2）如果用户还没有微博号，单击“注册”按钮进入注册界面，在注册界面填写好注册信息，单击“注册”按钮连接服务器进行注册。

（3）注册成功或者登录成功后，用户可以进入个人中心，个人中心包含了用户可以使用的功能和服务，在“快速发布”界面，用户可以选择“更新心情”、“发布日志”和“拍照上传”三个快速通道。在个人中心界面按下手机屏幕上的“MENU”按钮后，可弹出“搜索”和“退出”选项菜单。

（4）还可以查看自己的好友列表，好友列表中除显示好友的头像外，还将显示用户的昵称和心情。



（5）用户可以单击“最近访客”选项卡查看最近访问过自己微博的用户，访问列表中将列出访问者的头像、昵称和访问时间。

（6）在个人中心，用户可以查看自己的个人日志列表，个人日志列表中列出了日志的标题、缩略内容，用户可以单击每个日志右侧的“编辑”或者“删除”按钮对日志进行管理。

（7）单击个人中心的“相册列表”，用户可以管理自己的相册，包括查看相册照片和修改相册权限。

（8）在选项菜单中选择“搜索”选项，会打开搜索界面。在文本框中输入关键字，单击“搜索”按钮，服务器会将符合条件的微博用户的头像、昵称及心情列出。

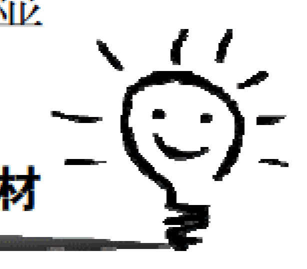
（9）在好友列表、访客列表和搜索界面的搜索到的用户列表中，单击列表中的某一项，都会转到该用户的主页界面，在该界面中用户可以查看并评论用户的日志，还可以按照设置的权限查看该用户的相册。

5. Android 端总体架构

在对微博随身的 Android 的功能进行了预览之后，下面对 Android 端的总体架构进行介绍。

对模块及其所包含的 Activity 类进行简单的介绍，这样读者就会对整个 Android 端的项目结构有一个总体的认识。

- ☑ 登录模块，由 LoginActivity 类实现，该 Activity 是程序运行后首先被启动的 Activity。注册模块由 RegActivity 实现，该 Activity 从 LoginActivity 启动。
- ☑ 个人中心模块，由 FunctionTabActivity 实现，该 Activity 从 LoginActivity 和 RegActivity 启动，其继承自 TabActivity，主要的功能是将用户的功能以选项卡的形式显示到屏幕上。
- ☑ 快速发布模块，由 PublistActivity 实现，从 FunctionTabActivity 启动，主要的功能是为用户提供发布日志、拍照上传和更新心情的功能选项。拍照上传模块由 ShootActivity 和 UploadActivity 实现，前者负责调用 Android 系统的照相机拍摄照片，后者负责将拍好的照片上传到用户的个人相册中。发布日志模块由 Publish DiaryActivity 实现，主要的功能是让用户编写新日志并发布。更新心情模块由 FunctionTabActivity 中的 AlertDialog 对象来实现。
- ☑ 管理个人相册模块，由 MyAlbumActivity 和 AlbumActivity 实现，前者负责显示个人相册列表，后者负责显示指定相册中的照片。管理个人日志模块由 MyDiary Activity 和 ModifyDiaryActivity 实现，前者负责显示用户个人的日志，后者负责向用户提供修改日志的功能。
- ☑ 查看联系人模块，由 ContactsActivity 实现，好友列表和访问列表均通过该 Activity 来显示。
- ☑ 搜索用户模块，由 SearchActivity 实现，其功能是接收用户输入的关键字，连接服务器进行查询，并将结果显示给用户。
- ☑ 博友注册模块，由 HomePageActivity、DiaryActivity、AlbumListActivity、Album Activity 和 CommentActivity 实现。HomePageActivity 继承自 TabActivity，负责显



示博友日志和博友相册两个选项卡。CommentActivity 由 DiaryActivity 启动，主要负责显示指定日志的评论列表，并且可以让用户发表新的评论。

除了以上实现特定功能的 Activity 之外，微博随身的 Android 端还包含一些公共的工具类，如 ConstantUtil 中封装了程序中会用到的全部常量；MyConnector 类提供了与服务器进行交互的 Socket 及输入/输出流对象。

小知识六：小窥 Web 前端开发

2005 年以后，互联网进入 Web2.0 时代，各种类似桌面软件的 Web 应用大量涌现，网站的前端由此发生了翻天覆地的变化。网页不再只是承载单一的文字和图片，各种富媒体让网页的内容更加生动，网页上软件化的交互形式为用户提供了更好的使用体验，这些都是基于前端技术实现的。

随着 Web2.0 概念的普及和 W3C 组织的推广，网站重构的影响力正以惊人的速度增长。XHTML+CSS 布局、DHTML 和 Ajax 像一阵旋风，铺天盖地席卷而来，包括新浪、搜狐、网易、腾讯、淘宝等在内的各种规模的 IT 企业都对自己的网站进行了重构。

为什么它们会对自己的网站进行重构呢？有两个方面的原因：

第一，根据 W3C 标准进行重构后，可以让前端的代码组织更有序，可显著改善网站的性能，还能提高可维护性，对搜索引擎也更友好。

第二，重构后的网站能带来更好的用户体验，用 XHTML+CSS 重新布局后的页面，文件更小，下载速度更快。

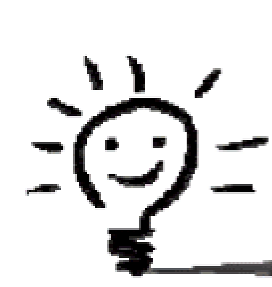
网站重构的目的仅仅是为了让网页更符合 Web 标准吗？不是。重构的本质是构建一个前端灵活的 MVC 框架，即 HTML 作为信息模型（Model），CSS 控制样式（View），JavaScript 负责调度数据和实现某种展现逻辑（Controller）。同时，代码需要具有很好的复用性和可维护性。这是高效率、高质量开发以及协作开发的基础。

DHTML 可以让用户的操作更炫，更吸引眼球；Ajax 可以实现无刷新的数据交换，让用户的操作更流畅。对于普通用户来说，一个网站是否专业、功能是否强大，服务器端是用 J2EE+Oracle 的强大组合，还是用 ASP+Access 的简单组合，并没有太明显的区别。但是，前端的用户体验却给了用户直观的印象。

随着人们对用户体验的要求越来越高，前端开发的技术难度越来越大，Web 前端开发工程师这一职业终于从设计和制作不分的局面中独立出来。

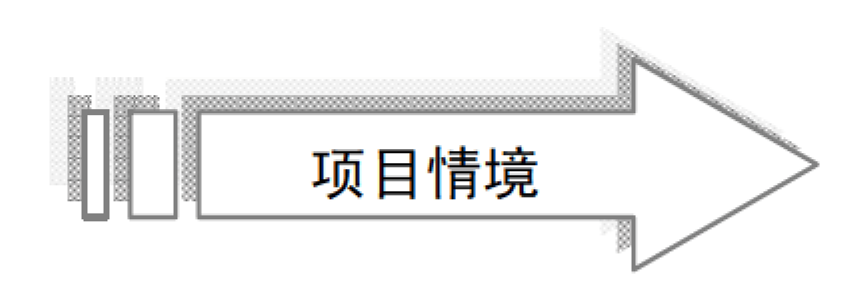
Web 前端开发技术包括三个要素：HTML、CSS 和 JavaScript，但随着 RIA 的流行和普及，Flash/Flex、Silverlight、XML 和服务器端语言也是前端开发工程师应该掌握的。Web 前端开发工程师既要与上游的交互设计师、视觉设计师和产品经理沟通，又要与下游的服务器端工程师沟通，需要掌握的技能非常多。这就从知识的广度上对 Web 前端开发工程师提出了要求。如果要精于前端开发这一行，也许要先精十行。然而，全才总是少有的。所以，对于不太重要的知识，我们只需要“通”即可。但“通”到什么程度才算够用呢？对于很多初级前端开发工程师来说，这个问题是非常令人迷惑的。

前端开发的入门门槛其实非常低，与服务器端语言先慢后快的学习曲线相比，前端开发的学习曲线是先快后慢。所以，对于从事 IT 工作的人来说，前端开发是个不错的切入点。

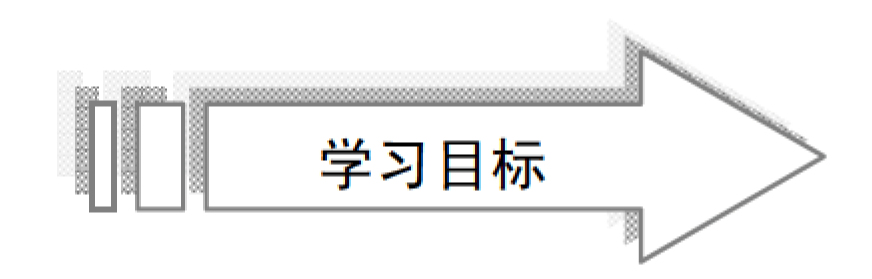


也正因为如此，前端开发领域有很多自学成“才”的同行，但大多数人都停留在会用的阶段，因为后面的学习曲线越来越陡峭，每前进一步都很难。另一方面，正如前面所说，前端开发是个非常新的职业，对一些规范和最佳实践的研究都处于探索阶段。总有新的灵感和技术不时闪现出来，如 CSS sprite、负边距布局、栅格布局等；各种 JavaScript 框架层出不穷，为整个前端开发领域注入了巨大的活力；浏览器大战也越来越白热化，跨浏览器兼容方案依然是五花八门。为了满足“高可维护性”的需要，需要更深入、更系统地去掌握前端知识，这样才可能创建一个好的前端架构，保证代码的质量。

项目六
地图设计器的开发



本项目介绍游戏地图设计器的开发，在该类游戏的开发中，地图设计器的使用是必不可少的。考虑到本游戏的特殊性，第三方地图设计器并不能满足需求，所以需要自行开发特定的地图设计器，下面将对战国英雄传游戏的地图设计器进行介绍，该地图设计器会为后面的游戏开发提供地图信息文件。



- 掌握底层地图设计器的开发方法，包括其设计思路、框架介绍和开发步骤
- 掌握上层地图设计器的开发方法，包括其框架介绍和开发步骤等

任务 11　底层地图设计器的开发

【任务情境】

本游戏中的地图有两层：底层全部是不可遇的实体，如道路、木桩以及花草等；上层主要是可遇的建筑，如城池、村庄、驿站等。接下来先对底层地图设计器进行开发。

【相关知识】

1. 地图设计器的开发设计思路

首先对地图设计器的设计思路进行简单介绍，使读者增加对该地图设计器的理解，其具体步骤如下。

（1）开发元素设计模块，该模块负责设计该层地图中所用到的所有元素，如道路、草地、木桩等，包括设计其占位点以及不可通过点。

（2）开发元素保存功能，能够将设计好的元素列表保存到指定文件中，下次重新启动设计器时可以直接加载保存过的元素列表，无需从头重新设计。

（3）开发层设计模块，该模块负责使用之前设计好的元素来设计地图的某一层，进入该模块前应该已经设计好所有元素并加载到元素列表中。

（4）开发层的保存功能，使用同样的技术将设计好的层信息保存到文件中，下次加载回来后可继续进行开发。

（5）地图信息文件的生成，产生包含该层所有信息的文件，该文件即为地图文件，将该文件存放到 Android 项目中的 assert 文件夹下，游戏可以直接读取其信息。

2. 底层地图设计器的框架介绍

底层地图设计器分为两部分：一部分是底层地图中元素的设计，包括设计占位点以及不可通过点；另一部分是层的设计，使用之前设计好的地图元素来设计底层地图。

3. 底层地图设计器的开发步骤

底层地图设计器的开发步骤如下。

（1）框架的搭建，首先创建一个普通的 Java 项目，然后搭建框架，如图 6-1 所示。

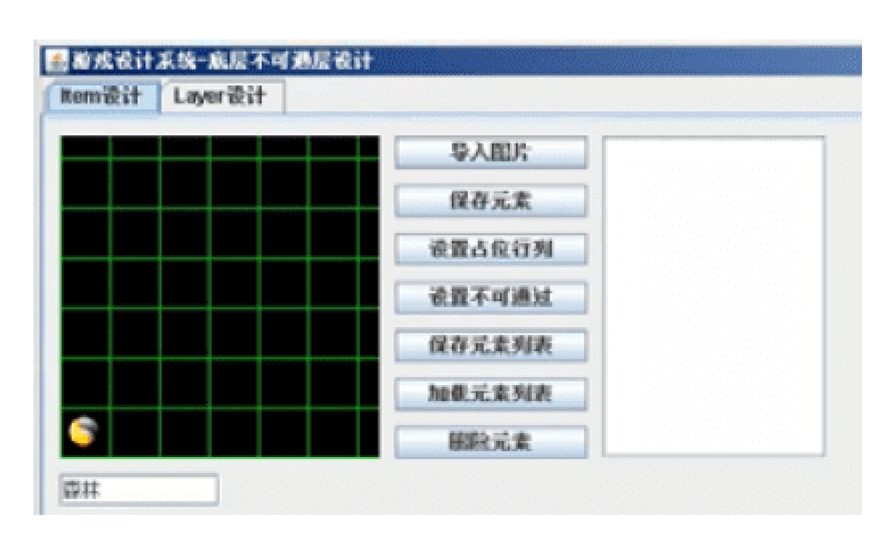


图 6-1　底层地图设计器框架搭建图

（2）开发“导入图片”按钮的事件处理，当用户单击“导入图片”按钮，应该弹出文件选择窗口，在文件选择窗口中选择一张图片，并将该图片显示到元素设计区域的左下角。

（3）开发“设置占位行列”以及“设置不可通过”按钮的监听事件，此时导入一张图片，便可通过按钮设置图片的占位点以及不可通过点，如图 6-2 所示。



图 6-2　设置图片占位点效果

（4）“保存元素列表”按钮事件的开发。根据用户设置的占位点以及不可通过图片的信息创建一个 Item 对象，并将该对象添加到元素列表中。

（5）之后开发保存元素列表的代码。当用户单击“保存元素列表”按钮时将元素列表序列化即可，前提是 Item 类实现了 Externalizable 接口并实现了接口中的方法。

保存的代码如下。

```
ArrayList<Item>allItem=new ArrayList<Item>();                    //成员变量，用于存放 Item
if(e.getSource()==jbSaveList){                                   //保存元素列表
    status=5;                                                    //将界面的状态值置成 5
    try{
        FileOutputStream fout=new FileOutputStream("ItemList.wyf"); //创建文件流
        ObjectOutputStream oout=new ObjectOutputStream(fout);    //序列化流
        oout.writeObject(allItem);                               //将元素列表 allItem 序列化
        oout.close();                                            //关闭相关流
        fout.close();
    }
    catch(Exception ea){                                         //捕获异常
        ea.printStackTrace();                                    //打印异常信息
    }
}
```

（6）“加载元素列表”按钮的事件处理代码的开发。当用户单击该按钮时，应该从文件 ItemList.wyf 中读取数据，直接恢复到元素列表 allItem，并将其显示到元素列表窗口中。

（7）元素模块的最后一步是开发删除元素功能，先得到元素列表中用户所选中的元素，然后将其从列表 allItem 中删除即可。

（8）搭建层设计界面，效果如图 6-3 所示。

（9）此时可以编写元素列表的事件监听方法。选中某个元素后，再单击右侧的设计窗口便将选中的元素添加到后台的 Item 数组中，并在右侧的窗口中显示。



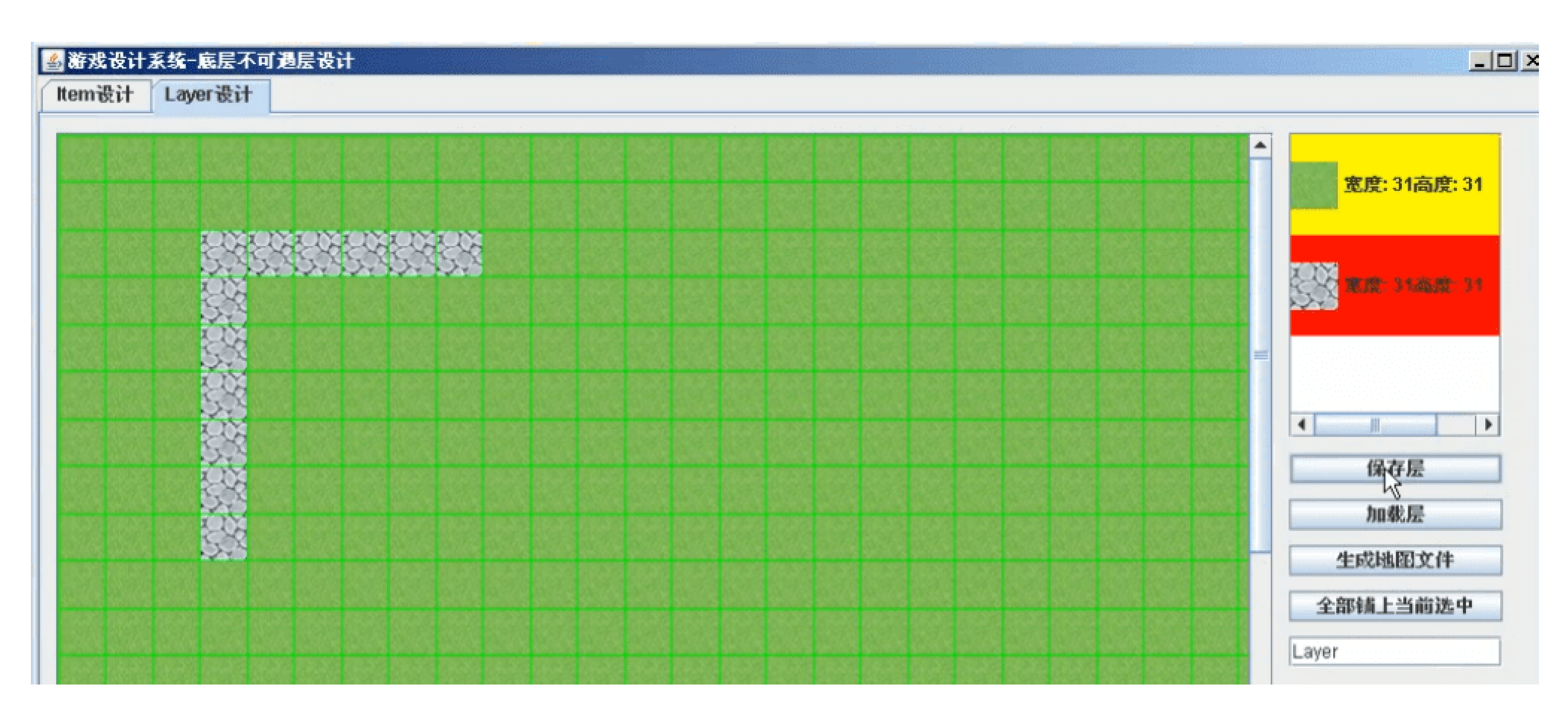


图 6-3　搭建层设计界面效果

（10）之后开发保存与加载的处理代码。保存时将 Item 的二维数组 itemArray 序列化到指定文件中，而加载时从保存的文件中读取 itemArray 并恢复界面的显示。

（11）“生成地图文件”按钮的处理，代码如下（将下列代码插入到按钮监听方法中）。

```
if(e.getSource()==jbCreate){                          //单击生成地图文件按钮
    try{                                              //捕捉异常
/*初始化输出流并计算地图数组中元素的总个数*/
FileOutputStream fout = null;                         //声明文件流
        DataOutputStream dout = null;                 //声明数据流
        fout = new FileOutputStream("maps.so");       //初始化文件流
        dout = new DataOutputStream(fout);            //初始化数据流
        int totalBlocks=0;                            //计数器
        for(int totalBlocks=0;
            for(int i=0;i<40;i++){
                for(int j=0;j<60;j++){                //循环
                    Item item = itemArray[i][j];
                    if(item != null){
                        totalBlocks++;                //计算有多少个 Item 对象
                    }
                }
            }
        dout.writeInt(totalBlocks);                   //写入不空块的数量
/*对地图数组进行循环，将数组每个 item 元素的信息写入到输出流中，即保存到文件中，然后将相关流关闭*/
for(int i=0;i<40;i++){
        for(int j=0;j<60;j++){                        //对地图数组循环
            Item item = itemArray[i][j];              //得到地图中该位置的元素
            if(item != null){
                int w = item.w;                       //元素的图片宽度
                int h = item.h;                       //元素的图片高度
                int col = item.col;                   //元素的地图列
```

```
                    int row = item.row;                             //元素的地图行
                    int pCol = item.pCol;                           //元素的站位列
                    int pRow = item.pRow;                           //元素的占位行
                    String leiMing = item.leiMing;                  //类名
                    int [][] notIn = item.notIn;                    //不可通过
                    int [][] keYu = item.keYu;                      //可遇矩阵
                    //计算图片下标
                    int outBitmapInxex=0;
                    if(leiMing.equals("Grass")){                    //是草地时
                         outBitmapInexe=0;
                    }else if(leiMing.equals("XiaoHua1")){           //是小花 1 时
                         outBitmapInexe=1;
                    }else if(leiMing.equals("MuZhuang")){           //是木桩时
                         outBitmapInexe=2;
                    }else if(leiMing.equals("XiaoHua2")){           //小花 2 时
                         outBitmapInexe=3;
                    }else if(leiMing.equals("Road")){               //是道路时
                         outBitmapInexe=4;
                    }else if(leiMing.equals("Jing")){               //是井时
                         outBitmapInexe=5;
                    }
                    dout.writeByte(outBitmapInexe);                 //记录图片下标
                    dout.writeByte(0);                              //记录可遇标志 0-不可遇
                    dout.writeByte(w);                              //图片宽度
                    dout.writeByte(h);                              //图片高度
                    dout.writeByte(col);                            //总列数
                    dout.writeByte(row);                            //总行数
                    dout.writeByte(pCol);                           //占位列
                    dout.writeByte(pRow);                           //占位行
                    int bktgCount=notIn.length;                     //不可通过点的数量
                    dout.writeByte(bktgCount);                      //写入不可通过点的数量
                    for(int k=0;k<bktgCount;k++){                   //写入不可通过矩阵
                         dout.writeByte(notIn[k][0]);
                         dout.writeByte(notIn[k][1]);
                    }
               }
          }
     }
     dout.close();                                                  //关闭数据流
     fout.close();                                                  //关闭文件流
}catch(Exception ea){                                               //捕获异常
     ea.printStackTrace();                                          //打印异常信息
}
}
```

任务 12　上层地图设计器的开发

【任务情境】

上层地图设计器与底层基本相同，只有可遇点的设置不同。

【相关知识】

上层设计器的元素设计界面完成后的效果如图 6-4 所示。



图 6-4　上层设计器元素设计界面

具体开发步骤如下。

（1）基于下层的元素设计界面开发上层，只需向底层元素界面中增添一个“设置可遇行列”按钮，当单击该按钮时，同样只需将可遇点存入数组中。

（2）上层设计界面比底层少一个“全部铺上当前选中”按钮，因为上层不会用同一元素填满整个地图，而下层一般需要先将整个地图铺满草地再进行设计。

（3）最后只需将保存的代码稍微修改一下，将生成的地图信息文件改为 mapsu.so，将计算图片下标的代码改为根据上层元素计算的下标，即将任务 11 代码中第 32～44 行用于判断 equals 的元素改成上层地图所拥有的元素名称。

到此，本游戏的地图设计器便开发完成，因为篇幅有限，只对地图设计器的开发思路进行了简单介绍，并没有给出详细的代码。

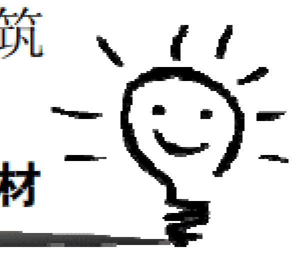
【项目小结】

本项目介绍了地图设计器的开发方法，该设计器将为后面的游戏开发提供地图信息文件，游戏中的地图是两层的，底层全部是不可遇的实体，上层地图设计器与底层基本相同，只有可遇点的设置不同。本项目详细介绍了两层的开发步骤，同样的方法可用于同类游戏地图设计器的设计和开发过程。

🕮 小知识七：游戏地图设计相关要素

☑　**构架**

背景：该地图的历史背景及相关事件，关系到整张地图的平面轮廓、地貌、特色建筑



区域、补给点、道路和任务链的设定。当然，也可以在设计好的地图基础上，补写故事背景，所以这一点不是必需的，只看游戏的侧重点及做法。

轮廓：该地图的平面轮廓形状。关系到地图的地貌，即河流、山脉的走向，湖泊、山丘的分布，从而关系到道路等一系列要素的设定。在无缝结合地图中影响到诸多地图的相互拼接，在分张地图上则影响不大。

地貌：高山、河流、湖泊、沙滩等地理环境的布置，影响到道路的设计、怪物的种类及分布区域、补给点位置及任务设计。

物件：树、草、顽石、建筑物、破烂物（破船）、尸骨遗物等物件的摆放，影响到地图的感观及任务的相关设计。

补给点：玩家休整地点，包括城市、村镇、NPC 聚集点等这些商业及任务的集合地，关系到玩家练级、任务的爽快感和便捷性，地理位置必须合理。

天气：阴、晴、雨、雪的变化，常驻或变换，根据引擎及需求设定，影响到地图的感观及玩家的体验。

☑　**区域划分**

整体地图：游戏的世界地图划分成诸多章节地图，可以看成一个个小国家、小势力，怪物等级跨越幅度大，地图构架区别显著。诸多章节地图有机拼接，形成一整张世界地图，但是拼接方式不同，一般分成两种。

（1）嵌接：就是两地图边缘轮廓形状相对应，紧密嵌入，形成无缝的接合。可以是无缝接合地图，也可以是分张地图游戏的世界地图。

（2）空间过渡：这样的世界地图比较特殊，每个章节地图之间有很大的空隙，并没有紧密相连。这些空隙可以允许玩家经过，也可以不允许，但是给人的感觉就是章节地图是摆放在世界地图上的一颗颗棋子。

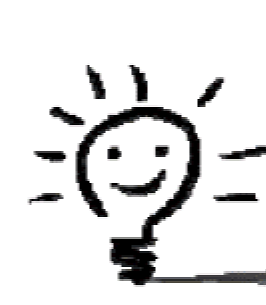
局部地图：把每一个章节地图按照顺序串联起来，形成一条伴随玩家成长的线，让玩家等级逐步提高，对世界背景的了解逐步加深，对系统逐渐熟悉。这是指总的地图布局，而每个章节地图，也应当有一条主线，外加多条支线，让玩家顺利成长到下一阶段，到达新地图，进行新的冒险活动。因此，根据玩家成长的需要，要把章节地图再细分成诸多片区。如何划分，可以以下面几条为依据。

（1）依据成长路线：如升级路线和体验世界路线，这些都是最重要的依据。为了引导玩家练级，以该地图的补给点为端点，将怪物由近至远地逐渐提高等级，增加职业组合，从而划分出不同的练级区域。例如，补给点在 A，由内到外，即 A 到 B 再到 C，那么 A 与 B 之间形成 1～5 级片区，B 与 C 之间形成 5～10 级片区。

（2）依据地形：按自然景观将地图分割出若干区域，而每个区域则要刷新符合当地背景的怪物及其他要素。

（3）依据怪物：某些特定怪物和 NPC 的聚集点，要把这些区域划分出来，形成独立的片区。例如，《魔兽世界》中西瘟疫之地的壁炉谷，这片区域独立于整张地图玩家成长路线存在，是有特殊意义的地点。

（4）依据任务：特殊精彩的任务要给予特定的区域，这条大多与第三条相结合，诸如各种复杂的任务链和庞大的副本，这些区域要独自成片。



☑ **线路设计**

概述：线路设计就是将地图中重要的建筑、NPC、怪物区域、任务区域、副本区域等一切我们想要让玩家看到的地图要素以最佳线路相连接，并作为玩家进行冒险的最直接路标。地图轮廓曲折、地貌各式各样，所以线路也就变得蜿蜒曲折，没了特定形状，这样我就很难说清楚每条线路设计的优势所在了，所以我在这里作出最基本的线路形状定义。

散射型：是最基本的线路。以补给点或地图中心地带为圆心，向四周扩散出去。全方位共 8 个方位，每个方位都可以独立成片，同一方位亦可分割成多个片区。可以有道路通向外围，也可无道路，让玩家自由行走。可以一马平川，亦可有山川阻碍。在某些特定地图中，散射型可能为局部散射，也就是方位数小于 8。这种形式的线路一般应用于小地图中，由于玩家需要从中心到边缘进行循环往复的奔走，所以过长的半径有碍于玩家的游戏体验。另外，根据其多样性，作出如下分类。

（1）全方位（8 个方位）

自由：以中心补给点向四周延伸，多用于平原地区。一般根据地形需求，左与右，上与下的延伸幅度可以不相同。

阻碍：以全方位自由散射为基础，在某些方位上会遇到大的阻碍，需要绕行。这种形式可以用于盆地、山地等区域。

（2）局部方位（小于 8 个方位）

自由：这种情形的中心补给点一般靠在山脉、海洋旁边，导致玩家不能够向山脉方位行进。

阻碍：在上述基础上，在可延伸方位上又出现了不同程度的阻碍，这多应用于较复杂的地形当中。

叶脉型：如同羽状叶脉，在主干道上留有若干节点，再在节点上延伸出支路。其实这种线路是在散射型基础上发展而来。散射型适应的地图的面积不宜过大，我们将各个散射型线路的中心点用主干道串连起来，就形成现在的叶脉型线路。这种线路可以适应各种较大的地图，面对各种地形也有较好的适应性，由于每个补给点到最远地图处的距离差不是很大，所以不会对玩家造成某些困扰。个人把它分成两种类型，而每种类型又有狭长与宽广之分。

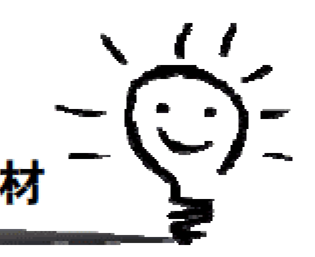
（1）直线多节点：主干道大体成一条直线形状，并在直线上设立若干节点，也就是补给点，在节点处发出多条支路到地图的各个区域。这种线路设计适合长条形地图，如果地图很长，我们就多设置补给点，这样玩家就不会感受到过多奔走带来的烦恼。

狭长型：根据地形地貌的需要，会出现很狭长的地图，这样玩家横向移动范围很小，会浪费地图的纵深度。当然，如果背景设定有需求，这是必须要做的，但是不宜太多，能避免则尽量避免。

宽广型：玩家横向移动范围比较大，有纵深度，是较为常用的类型。

（2）曲线多节点：主干道为一条曲线，这主要是为了配合比较复杂的地形地貌而设计的，毕竟地图的轮廓不可能都是规矩的长条形。在某些很规整的地图中，也会用到曲线型的主干道，例如，《魔兽世界》的东瘟疫地图。

狭长型：同上。



宽广型：同上。

圆环型：主干道形成一个环状，可以看成是曲线叶脉型，首尾相连接而成。可以用于大地图中，但是补给点要设置妥当，最少也要 2 个。环中心可以是可通过的平原，这种情况又可以看成很大的散射型线路。如果中心为不可通过的高山、湖泊之类的场景，那么通过环状的线路，也不会让玩家绕远路。

依托地貌不规则型：某些地图的地形过于特殊，例如，出现了大裂谷、钳口形海滩等大面积的不规则形状场景，线路设计则只能依据地貌进行设计，从而导致线路外形来回扭曲不断，或分支线路相当长且不规则。

网格型：有些地图的建筑布局比较规整，所以线路也就比较规则，交错成网状。以《天堂 2》的世界地图为例，它的每个章节地图都均匀地排列在大地图上，而且间隔距离差距不大，这也就可以用网格线将这些章节地图一一圈起来，于是这网格线就成了游戏的主要道路。而在某些特殊的杀怪片区，例如，怪物的城市中，既然是城市，那么道路就会平直，并且相互交错，也就成为了网格型线路。

独立空间连接型：如果说一个宽广的场所是一个房间，那么一条狭长的通路就是走廊，这样走廊就将大量的房间相连接，从而形成一个独具特色的地图。这种多用于副本地图，也可做成迷宫，《征途》中的一些洞穴地图就用到了这种设定。这种线路根据形状可以分为两种类型。

通常：多个宽广的房间由狭长长廊链接。

变形：长廊变短、变宽，甚至消失，直接由房间与房间相连，只以门为界。

螺旋型：立体型的线路，例如，高塔、高楼、高山等建筑的通路，或者低洼地等凹陷场景的通路，一般很少用到。对章节的图片区连接没有太大的影响。

☑　**怪物设置**

概述：关卡离不开怪物的配合，或者说设计的关卡要符合怪物的特性，包括技能和移动等。只有更好地发挥出怪物的威力，才能够给予关卡更多的难度和变化性，也能减少关卡的复杂度，为设计者带来方便。

☑　**举例**

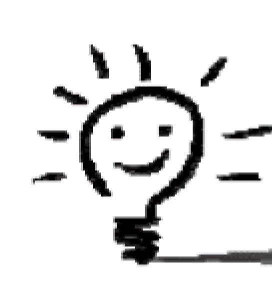
在大范围的近战怪物中添加几个远战怪物，就可大幅度增加难度。

在大范围的物理型怪物中添加少量魔法型怪物，就可大幅度增加难度，并且增加玩家的点券消耗，如《征途》中的几个山洞就有物理和魔法，而玩家想要挂机就需要准备双防装备。

综合实训六　微博随身之 Web 端开发

【问题情境】

前面已经介绍了微博随身 Web 端的主要功能和系统架构，本实例会对 Web 端各个功能模块的开发进行具体介绍。



【拓展知识】

1．Web 端主页的搭建

下面首先介绍 Web 端主页的搭建。

微博随身 Web 端的主页对应的 JSP 文件为 index.jsp。主页主要分为 4 个部分，分别用来显示发布页面、登录注册页面、好友及访客页面及信息内容页面。index.jsp 的主要组成结构是<iframe>，其代码如下。

```
<%@ page language="java" contentType="text/html; charset=GBK"
    pageEncoding="GBK" import="wpf.DBUtil,java.sql.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
                        "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <LINK href="global.css" type="text/css" rel="stylesheet"/>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=GBK"/>
        <title>口袋微博</title>
    </head>
    <body background="img/back.jpg">
    <br/><br/><br/><br/><br/><br/><br/>
        <table class="bodyBack" width="80%" border="0"align="center"
            style="overflow:auto;height:e-xpression(document.body.offsetHeight);
            padding: 0em; margin: 0em;" cellspacing="0" cellpadding="0">
            <tr>
                <td align="left" valign="top" bgcolor="#9c9c9c" width="600px" style="
                padding: 0em; margin: 0em;" class="bodyBack"> <!--包含发布页面的 iframe-->
              <iframe name="write" src="write.jsp"   width="100%" frameborder="0"
                        scrolling="no" height="185px" > </iframe>
              </td>
              <td align="center" valign="bottom" width="258px"><!--包含登录页面的 iframe -->
                 <iframe id="login" name="login" width="100%" src="login.jsp"
                   frameborder="0" scrolling="no"></iframe>
             </td>
        </tr>
        <tr>
              <td height="100%"><!--包含不同内容信息页面的 iframe -->
                 <iframe name="content" width="100%" height="800px" scrolling="no"
                 frameborder="0"></iframe>
              </td>
              <td height="100%"> <!--包含好友和访客列表页面的 iframe -->
                 <iframe id="friend" name="friend" width="100%" scrolling="auto"
                 height="800px" src="contacts.jsp" frameborder="0"></iframe>
              </td>
        <tr>
        </table>
    </body>
</html>
```


说明

index.jsp 的页面主要通过表格来定位，每个<td></td>标记之间放一个<iframe>标记，其中名为“write”的 iframe 将显示发布页面；名为“login”的 iframe 将显示登录/注册页面；名为“friend”的 iframe 将显示好友和访客页面；名为“content”的 iframe 将根据不同的请求显示不同的内容信息，如相册、日志等。

2. Web 端登录注册模块的实现

打开主页之后，无论是使用微博随身平台发布日志或照片，还是查看其他用户分享的内容信息，首先都需要登录系统，如果用户还没有微博号，就需要进行注册获取微博号后再进行登录。下面介绍 Web 端登录注册模块的开发。

（1）用户登录功能的开发

下面将介绍用户登录功能的开发。打开 Web 端的主页后，主页右上角即是登录页面，登录页面的显示是通过 login.jsp 来实现的，通过向控制器 MyServlet 发出请求并处理其返回信息来实现已注册用户的登录。具体开发步骤如下。

① 开发 MyServlet、Myservlet 是微博随身 Web 端主要的控制器之一，几乎所有页面的请求都要通过 MyServlet 的处理和跳转。MyServlet 位于 Web 端项目 WEB-INF/classes 目录下，其代码如下。

```
package wpf;                                            //声明包语句
import static wpf.ConstantUtil.*;                       //引入相关类
import java.io.IOException;                             //引入相关类
…                                                       //此处省略部分引入相关类的代码
import javax.servlet.http.HttpServletResponse;          //引入相关类
import javax.servlet.http.HttpSession;                  //引入相关类
public class MyServlet extends HttpServlet {
    public MyServlet() {                                //构造器
        super();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {      //重写 doGet 方法
            doPost(request, response);                  //调用 doPost 方法
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {              //重写 doPost 方法
    request.setCharacterEncoding(CHAR_ENCODING);        //设置编码格式
    String action = (String)request.getParameter("action");     //获取 action
    if(action.equals("login")){                         //action 为登录信息
      String u_no = (String)request.getParameter("u_no");       //读取密码参数
    String u_pwd = (String)request.getParameter("u_pwd");       //读取密码参数
    ArrayList<String> result = DBUtil.checkLogin(u_no, u_pwd); //查询数据库
```

```
23                if(result.size()>1){                    //如果列表长度大于 1，表示登录成功
24                    HttpSession session = request.getSession();
25                    String no = result.get(0);          //获得用户的号码
26                    String name = new Sring(result.get(1).getBytes("ISO-8859-1"), CHAR_ENCODING);
27                                                        //获得用户的昵称
28                    String email = result.get(2);       //获取用户电子邮件
29                    String state = new String(result.get(3).getBytes("ISO-8859-1"), CHAR_ENCODING);
                                                          //获取用户状态
30                    String hid = result.get(4);         //获取用户头像
31                    User user = new User(no, name, email, state, hid);  //根据用户信息创建 User 对象
32                    session.setAttribute("user", user); //将用户 id 加入 Session
33                }
34                else{                                   //如果登录失败
35                    request.setAttribute("loginResult", result.get(0));          //将出错信息返回
36                }
37                request.getRequestDispatcher("login.jsp").forward(request, response);     //跳转到 login.jsp
38            }
39            …//此处省略了对于其他 action 的请求处理，将在后面的介绍中给出详细代码
40        }
41    }
```

☑ 第 11～14 行为重写的 doGet 方法，该方法将直接调用 doPost 方法处理请求。

☑ 第 15～39 行为重写的 doPost 方法，该方法将处理所有来自页面的请求并根据请求中 action 值的不同进行不同的处理。

☑ 第 17 行设置了编码格式，防止出现乱码。第 18 行首先获得请求中 action 参数的值，然后根据其值调用 DBUtil 工具类的不同方法完成浏览器的请求。

☑ 第 19 行判断 action 参数的值，如果为 login，说明是用户的登录请求，则获取 request 对象中的用户名和密码，调用 DBUtil 的 checkLogin 方法验证用户的登录信息。如果登录成功，则返回一个 User 对象，如果登录失败，则返回失败信息。User 类中封装了用户头像、昵称等信息，由于其代码很简单，在此不予列出。

提示

代码第 40 行省略对于其他 action 请求的处理代码，在后面介绍其他模块时将会详细地介绍。由于篇幅有限，在后面介绍 MyServlet 代码时，将只列举具体的 action 处理的代码片段，而不会再将 MyServlet 类的其他代码重复列出。

开发完 Servlet 后，还需要对其进行配置，打开项目 WEB-INF 目录下的 web.xml，在“</web-app>”标记之前插入如下配置代码。

```
<servlet>
    <display-name>MyServlet</display-name><!--Servlet 的显示名称-->
    <servlet-name>MyServlet</servlet-name><!--Servlet 的名称-->
```




```
        <servlet-class>wpf.MyServlet</servlet-class><!--Servlet 的全称类名-->
    </servlet>
    <servlet-mapping>
        <servlet-name>MyServlet</servlet-name><!--Servlet 的名称-->
        <url-pattern>/MyServlet</url-pattern><!--映射到该 Servlet 的 URL-->
    </servlet-mapping>
```

提示

如果使用 Eclipse for Java EE 开发环境，可以在新建文件时选择新建一个 Servlet，这样 Eclipse 会在 web.xml 中自动配置 Servlet。

② 开发 DBUtil，DBUtil 为系统的工具类，提供了包括连接数据库在内的静态业务方法来完成具体的功能，不仅 Web 端会用到 DBUtil 类中的方法，后面会介绍的微博随身 Android 手机版的服务器也会调用 DBUtil 中的静态方法，其代码如下。

```
package wpf;                                          //声明包语句
import static wpf.ConstantUtil.*;                     //引入相关类
…                                                     //此处省略部分引入相关类的代码
import java.util.ArrayList;                           //引入相关类
public class DBUtil {
 public static Connection getConnection(){            //方法：使用数据源连接池访问数据库
        Connection con = null;
        try{
            Context initial = new InitialContext();   //创建 context 对象
            DataSource ds = (DataSource)initial.lookup("java:comp/env/jdbc/mysql");
                                                      //获得数据源
            con=ds.getConnection();
        }
        catch(Exception e){                           //捕获打印异常
            e.printStackTrace();
        }
public static ArrayList<String> checkLogin(String u_no,String u_pwd){
                                                      //方法：检查用户是否登录成功
        ArrayList<String> result = new ArrayList<String>();
        Connection con = null;                        //声明获取数据库连接
        PreparedStatement ps = null;                  //声明 Statement 对象
        ResultSet rs = null;                          //声明 ResultSet 对象
        try{
            con = getConnection();                    //获取数据库连接
            if(con == null){                          //判断数据库连接对象是否
                result.add(CONNECTION_OUT);
                return result;
            }
            ps = con.prepareStatement("select u_no,u_name,u_email,u_state,h_id from user
                         where u_no=? and u_pwd=?"); //创建预编译语句
            ps.setString(1, u_no);                    //设置预编译语句的参数
```

```
30              ps.setString(2, u_pwd);                      //设置预编译语句的参数
31              rs = ps.executeQuery();
32              if(rs.next()){                               //判断结果集是否为空
33                  for(int i=1;i<=5;i++){
34                   result.add(rs.getString(i));            //将结果集中数据存放到 ArrayList 中
35                  }
36              }else{                                       //如果数据库查无此人
37                  result.add(LOGIN_FAIL);                  //返回登录出错信息
38              }
39          }catch(Exception e){e.printStackTrace();}        //捕获并打印异常
40          finally{
41              try{if(rs != null){rs.close(); rs = null;}}  //关闭 ResultSet 对象
42              catch(Exception e){e.printStackTrace();}
43              try{if(ps != null){ps.close(); ps = null;}}  //关闭 PreparedStatement 对象
44              catch(Exception e){e.printStackTrace();}
45              try{if(con != null){con.close();con = null;}} //关闭 Connection 对象
46              catch(Exception e){e.printStackTrace();}     //捕获并打印异常
47             }
48          return result;
49         }
50  ...//此处省略 DBUtil 类中其他业务方法的代码，将在介绍相应功能模块时进行详细介绍
51  }
```

☑ 第 6～15 行为 getConnection 方法，该方法的主要功能是通过配置好的数据源获取数据库连接对象 Connection，DBUtil 类中的其他工具方法都将调用该方法获得数据库连接。

☑ 第 10 行为获取数据源的代码，其中 mysql 为数据源的 JNDI 名称。

☑ 第 16～49 行为 checkLogin 方法，该方法的主要功能是根据传入的用户 ID 和密码进行登录验证，如果登录成功，返回该用户的相关信息，否则返回错误信息。

提示

> 代码第 50 行省略了 DBUtil 类中其他业务方法的代码，本书随后的介绍中将逐步完善 DBUtil 类，对这些业务方法进行介绍。

（2）用户注册功能的开发

用户在打开主页时除了进行登录之外，还可以选择注册新用户。下面将简单介绍用户注册功能的开发。该功能主要由前台页面 register.jsp、MyServlet 和 DBUtil 中的代码来实现。

① 当用户在注册页面填好注册信息并提交后，由 MyServlet 来接收并处理请求，MyServlet 类中处理用户注册请求的代码片段如下。

```
else if(action.equals("register")){                            //action 为注册信息
  String u_name = (String)request.getParameter("u_name");      //接收用户昵称参数
  String u_pwd = (String)request.getParameter("u_pwd");        //接收用户密码参数
  String u_email = (String)request.getParameter("u_email");    //接收用户邮箱地址参数
```

```
5    String u_state = (String)request.getParameter("u_state");        //接收用户心情参数
6    String result = DBUtil.registerUser(u_name, u_pwd, u_email, u_state,"0"); //调用 DBUtil 类的方法
7    if(!result.equals(REGISTER_FAIL)){                                //注册成功
8        User user = new User(result, u_name, u_email, u_state, "0");          //创建 User 对象
9        HttpSession session = request.getSession();                   //获得 Session
10       session.setAttribute("user", user);                           //将 User 对象存放到 Session 中
11   }
12   request.setAttribute("result", result);                           //设置 request 的属性
13   request.getRequestDispatcher("register.jsp").forward(request,response);      //调转到 register.jsp
14   }
```

☑ 上述代码为 MyServlet 类中处理用户注册请求的代码，首先从 request 请求中获取用户的注册信息，然后调用 DBUtil 类的 registerUser 方法进行注册。

☑ 第 7～11 行为判断是否注册成功的代码，如果注册成功，则创建一个 User 对象并将其存放到 Session 中，否则将信息直接返回。

② 在上述代码中调用了 DBUtil 类的 registerUser 方法，其代码如下。

```
public static String registerUser(String u_name,String u_pwd,String u_email,String u_state,String
h_id){
        String result=null;
        Connection con = null;                  //声明数据库连接对象
        PreparedStatement ps = null;            //声明语句对象
        try{
            con = getConnection();
            if(con == null){                    //判断是否成功获取连接
                result = CONNECTION_OUT;
                return result;                  //返回方法的执行
            }
            ps = con.prepareStatement("insert into user(u_no,u_name,u_pwd,u_email, u_state,
            h_id)" + "values(?,?,?,?,?,?)");    //构建 SQL 语句
            String u_no = String.valueOf(getMax(USER));      //获得分配给用户的微博号
            u_name = new String(u_name.getBytes(),"ISO-8859-1");
                                                //转成 ISO-8859-1 以插入数据库
            u_state = new String(u_state.getBytes(),"ISO-8859-1");
                                                //转成 ISO-8859-1 以插入数据库
            int no = Integer.valueOf(u_no);
            int hid = Integer.valueOf(h_id);
            ps.setInt(1, no);                   //设置 PreparedStatement 的参数
            ps.setString(2, u_name);            //设置预编译语句中 u_name 字段的参数
            ps.setString(3, u_pwd);             //设置预编译语句中 u_pwd 字段的参数
            ps.setString(4, u_email);           //设置预编译语句中 u_email 字段的参数
            ps.setString(5, u_state);           //设置预编译语句中 u_state 字段的参数
            ps.setInt(6,hid);
            int count = ps.executeUpdate();     //执行插入语句
            if(count == 1){                     //如果数据插入成功
```

```
26                  result = u_no;                          //获得玩家的账号
27              }
28              else{                                       //如果没有插入数据
29                  result = REGISTER_FAIL;                 //获得出错信息
30              }
31          }catch(Exception e){e.printStackTrace();}
32          finally{
33              try{if(ps != null){ps.close(); ps = null;}  //关闭预编译语句
34              }catch(Exception e){e.printStackTrace();}
35              try{if(con != null){con.close();con = null;} //关闭数据库连接
36              }catch(Exception e){e.printStackTrace();}
37          }
38          return result;                                  //返回方法执行结果
39      }
```

- ☑ 第 6 行调用 getConnection 方法获得数据库连接，第 11 行创建预编译语句对象 preparedStatement，并为预编译语句设置参数。需要注意的是，在设置预编译语句参数时如果字段中包含中文，需要对其进行转码，否则会出现乱码现象。
- ☑ 第 13 行调用了 DBUtil 类的 getMax 方法获取新用户的微博号，用户注册成功后会将该微博号返回到页面，该微博号将用来登录到微博平台。
- ☑ 第 24 行为执行预编译语句的代码，第 25～30 行为判断数据插入是否成功的代码，如果 preparedStatement 对象的 executeUpdate 方法返回 1，则表示插入成功，否则失败。
- ☑ 第 32～37 行为关闭预编译语句和数据库连接对象的代码，将这些代码放到 finally 语句中是为了确保其一定被关闭。

③ 在上述 registerUser 方法中代码第 13 行调用了 DBUtil 的 getMax 方法，该方法不仅为用户提供新的微博号，还为数据库中其他以编号为主键的表提供了新插入记录的主键值。每当需要向这些表中插入新记录时，首先调用该方法，传入自己的表名并将方法的返回值作为新插入记录的主键值。

数据库中所有表的最大编号存放在 max 表中，该表中只有一条记录，不同的字段值代表不同表当前的最大编号，该条记录在数据表被创建后插入。

getMax 方法为同步方法，这样可以避免两个同时注册的用户具有相同微博号现象的发生，同时使用 getMax 方法获得编号还可以避免不同的用户先后使用同一个微博号码的情况发生。getMax 方法的代码如下。

```
public static synchronized int getMax(String table){
                                        //方法：获取指定表中的当前最大编号，该方法为同步方法
        int max = -1;                   //声明用户返回的整型变量
        Connection con = null;          //声明数据库连接对象
        Statement st = null;            //声明 Statement 对象
        ResultSet rs = null;            //声明 ResultSet 对象
        try{
            con = getConnection();      //获取数据库连接
```




```
8                   st = con.createStatement();                 //创建一个 Statement 对象
9                   String sql = "update max set "+table+"="+table+"+1";
10                  st.executeUpdate(sql);                      //更新最大编号
11                  rs = st.executeQuery("select "+table+" from max"); //查询最大编号
12                  if(rs.next()){                              //判断结果集中是否有数据
13                      max = rs.getInt(1);                     //获得当前最大值
14                      return max;
15                  }
16              }catch(Exception e){e.printStackTrace();}       //捕获并打印异常
17              finally{
18                try{
19                  if(rs != null){rs.close(); rs = null;}      //关闭结果集 ResultSet
20                  }catch(Exception e){e.printStackTrace();}   //捕获并打印异常
21                try{
22                  if(st != null){st.close(); st = null;}      //关闭语句对象
23                  }catch(Exception e){e.printStackTrace();}   //捕获并打印异常
24                try{
25                  if(con != null){con.close(); con = null;}   //关闭数据库
26                  }catch(Exception e){e.printStackTrace();}   //捕获并打印异常
27              }
28              return max;                                     //返回指定表的最大编号
29          }
```

☑ 第 9 行定义了一个 SQL 语句，该语句的功能是将表中指定的字段值加 1，getMax 方法接收的参数是一定的，是代表 max 表中各个字段的静态变量，来自 Constant Util 类。

☑ 第 10 行代码的功能是执行第 9 行创建的 SQL 语句。第 11 行代码是将刚更新的字段值检索出来，并将其作为 getMax 方法的返回值。

提示

> 注册模块中负责前台显示的 register.jsp，由于篇幅有限，将不会列出该 JSP 文件的代码。

（3）用户注销功能的开发

用户在打开首页时，可以选择登录还是注册，如果选择登录并且成功登录后，或者选择注册并且成功注册后，页面中会显示“注销”超链接。单击“注销”超链接后同样会向 MyServlet 发出 action 为“logout”的请求，MyServlet 处理该请求的代码如下。

```
else if(action.equals("logout")){                                    //action 为注销登录
        HttpSession session = request.getSession();                  //获得 Session 对象
        session.setAttribute("user", null);                          //将 user 属性设置为 null
        request.getRequestDispatcher("login.jsp").forward(request,response); //跳转到 login.jsp 页面
        }
```

说明

上述代码比较简单，当 MyServlet 收到动作为注销的请求时，就将 Session 中的 user 属性设置为 null，并重新跳转回 login.jsp 页面。

3. 查看和管理日志模块的实现

日志的发布和管理是微博随身中一个重要的功能，下面就来介绍日志管理模块的设计与实现。本模块包含的功能主要有发布日志、编辑日志、删除日志和评论日志。

（1）发布日志功能的开发

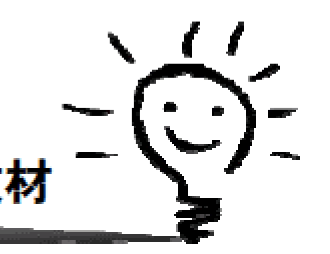
用户登录系统后，在主页的发布页面即可发布新的日志或更新心情，发布页面是由 write.jsp 负责显示的，用户在 write.jsp 页面中填写好新日志的标题和内容后，单击“发布”按钮，将向 MyServlet 发出请求，MyServlet 处理该请求的代码如下。

```
1    else if(action.equals("new_diary")){                    //action 为写新日记
2            HttpSession session = request.getSession();     //获取 Session
3            User user = (User)session.getAttribute("user");//获得 Session 中的 User 对象
4            if(user == null){                               //用户没有登录
5                request.setAttribute("result", DIARY_FAIL);
6                request.getRequestDispatcher("write.jsp").forward(request, response);
                                                             //跳转到 write.jsp
7                return;
8            }
9            String u_no = user.u_no;                        //获得用户的 id
10           String title = (String)request.getParameter("title");         //获得日志的标题
11           String content = (String)request.getParameter("content"); //获得日志内容
12           String result = DBUtil.writeNewDiary(title, content, u_no);
                                                             //调用 DBUtil 类的方法写入新日志
13           request.setAttribute("result", result);         //将结果设置到 request 的属性中
14           request.getRequestDispatcher("write.jsp").forward(request, response);//跳转返回
15   }
```

☑ 第 2～3 行为获取 Session 中的 User 对象的代码，如果 Session 中该对象不为空，则说明用户已登录；如果为空，说明用户已注销或未登录。

☑ 第 9～14 行为当用户已登录时对日志进行发布的代码，首先从 request 中读取日志的标题和日志的内容，然后调用 DBUtil 类中的 writeNewDiary 方法向数据库中写入新的数据。最终将 writeNewDiary 方法的返回值设置到 request 中的属性并跳转到 write.jsp。

上述代码的第 12 行调用了 DBUtil 类的 writeNewDiary 方法将新日志数据插入到数据库中，该方法的代码如下。



```
1   public static String writeNewDiary(String title,String content,String author){//
2           String result = null;
3           Connection con = null;
4           PreparedStatement ps = null;
5           try{
6               con = getConnection();                    //获得连接
7               ps = con.prepareStatement("insert into diary
8               (r_id,r_title,r_content,u_no)" +" values(?,?,?,?)");
9               int max = getMax(DIARY);                  //获取当前最大编号
10              ps.setInt(1, max);                        //设置各个字段的值
11              ps.setString(2, new String(title.getBytes(CHAR_ENCODING),"ISO-8859-1"));
12              ps.setString(3, new String(content.getBytes(CHAR_ENCODING),"ISO-8859-1"));
13              int u_no = Integer.valueOf(author);       //转成字符串
14              ps.setInt(4, u_no);
15              int count = ps.executeUpdate();           //更新数据库
16              if(count == 1){
17                  result = DIARY_SUCCESS;               //返回值
18              }
19              else{
20                  result = DIARY_FAIL;                  //设置返回值
21              }
22              return result;
23          }catch(Exception e){e.printStackTrace();}     //捕获并打印异常
24          finally{
25              try{if(ps != null){ps.close(); ps = null;}    //关闭预编译语句
26              }catch(Exception e){e.printStackTrace();}     //捕获并打印异常
27              try{
28                  if(con != null){con.close();con = null;}  //关闭数据库连接对象
29              }catch(Exception e){e.printStackTrace();}     //捕获并打印异常
30          }
31          return result;                                //返回发布日志的结果
32      }
```

- ☑ 第 6 行调用 getConnection 方法获得数据库连接，第 7～8 行创建了一个用于向 diary 表插入新记录的预编译语句。
- ☑ 第 9 行调用了 getMax 方法获取新日志的主键编号。第 10～14 行分别为预编译语句的各个参数赋值，需要注意的是需要对包含中文的字段值进行转码。CHAR_ENCODING 是 ConstanUtil 类中的静态变量，代表除 MySQL 数据库之外 JSP 页面和 Java 文件的编码格式。
- ☑ 第 15 行调用 preparedStatement 的 executeUpdate 方法向数据库的 diary 表插入新记录。代码第 16～21 行根据 executeUpdate 的返回值设置 writeNewDiary 方法的返回值。

（2）显示日志及评论功能的开发

通过 write.jsp 页面发布的日志可以在 diary.jsp 页面中查看，diary.jsp 除了负责显示用户的日志之外，还会显示对该日志的评论。diary.jsp 页面接收被访问用户的 ID 作为参数，通过调用 DBUtil 中的 getUserDiary 方法来获得日志，该方法的代码如下。

```
public static ArrayList<Diary> getUserDiary(String u_no,int currentPage, int span){
                                                      //方法：查询用户日记列表
        ArrayList<Diary> result = new ArrayList<Diary>();
        Connection con = null;                        //声明数据库连接对象
        Statement st = null;                          //声明语句对象
        ResultSet rs = null;                          //声明结果集对象
        int start = (currentPage-1)*span;             //计算起始位置
        String sql = "select diary.r_id,diary.r_title,diary.r_content,
          date_format(diary.r_date,'%Y-%c-%e %k:%i:%s'),diary.u_no,user. u_name" +
          " from diary,user where diary.u_no="+u_no+" and diary.u_no=user.u_no" +
          " order by diary.r_date desc" + " limit "+start+","+span;     //构建语句对象
        try{
            con = getConnection();                    //获得连接库连接
            st = con.createStatement();               //创建语句对象
            rs = st.executeQuery(sql);                //执行查询
            while(rs.next()){                         //读取结果集生成日志对象
                String rid = rs.getInt(1)+"";
                String title = new String(rs.getString (2).getBytes("ISO-8859-1"),
                CHAR_ ENCODING);
                String content = new String(rs.getString(3).getBytes("ISO-8859-1"),
                CHAR_ ENCODING);
                String date = rs.getString(4);        //得到时间
                String uno = rs.getInt(5)+"";
                String uname = new String(rs.getString(6).getBytes("ISO-8859-1"),CHAR_
                ENCODING);
                Diary d = new Diary(rid,title, content, uname, uno, date);
                result.add(d);
            }
            for(Diary d:result){                          //为每个日志生成评论列表
              ArrayList<Comments> cmtList = getComments(d.rid);     //查询评论列表
              d.setCommentList(cmtList);                  //将评论列表设置到对应的日志列表
            }
        }catch(Exception e){e.printStackTrace();}         //捕获并打印异常
        finally{
            try{if(rs != null){rs.close();rs = null;}     //关闭结果集对象
            }catch(Exception e){e.printStackTrace();}     //捕获并打印异常
            try{if(st != null){st.close();st = null;}     //关闭语句对象
            }catch(Exception e){e.printStackTrace();}     //捕获并打印异常
            try{if(con != null){con.close();con = null;}  //关闭数据库连接
            }catch(Exception e){e.printStackTrace();}     //捕获并打印异常
        }
        return result;                                    //返回查询到的结果
    }
```

☑ getUserDiary 方法是可以执行带分页效果的查询的，其传入的参数 currentPage 代表所要获取的第几页的内容，span 参数指定的是每页日记的个数。在本系统中，span 被设置为 1，即每页只显示一条日志，修改 span 的值可以改变每页显示的日记个数。

- ☑ 第 2～5 行声明了 getUserDiary 方法中所用到的和最终返回的对象应用，第 6 行代码通过传入的 currentPage 和 span 参数计算出本次查询从整个日志列表中的哪个位置开始获取日志。
- ☑ 第 7～10 行代码创建了一个用于查询 SQL 语句，该 SQL 语句将检索 user 表和 diary 表，将日志标题、日志内容、日志发布时间、日志所属者 ID 和日志所属者昵称等字段的值检索出来，并通过 limit 子句从全部结果中截取指定的一部分日志。
- ☑ 第 12～14 行为执行查询的代码，第 15～24 行为遍历结果集 ResultSet 并根据读取到的字段信息创建 Diary 对象的代码。Diary 类中封装了诸如日志标题、内容等信息，同时还包括一个用于存放评论的 ArrayList。
- ☑ 第 25～28 行为获取日志的评论的列表的代码，对于每一篇已添加到日志列表中的日志，都将调用 DBUtil 类的 getComments 方法获取其中评论列表，并通过 Diary 类的 setCommentList 将评论列表设置到 Diary 对象中。

提示

代码第 8 行的 SQL 语句中使用了 date_format 函数，该函数可以让数据库中的日期时间以自定义的格式字符串返回。

上述 getUserDiary 方法的代码中调用了 DBUtil 类的 getComments 方法，该方法的功能是查询指定日志的评论列表，其代码如下。

```
public static ArrayList<Comments> getComments(String r_id){//方法：获得指定日志的评论列表
        ArrayList<Comments> result = new ArrayList<Comments>();
        Connection con = null;                          //声明 Connection 对象
        PreparedStatement ps = null;
        ResultSet rs = null;
        String sql = "select date_format(comment.c_date,'%Y-%c-%e %k:%i:%s'),//创建 SQL 语句
                comment.c_content,user.u_name,comment.u_no" +
                " from comment,user where comment.r_id=? and
                user.u_no=comment.u_no order by comment.c_date desc";
        try{
            con = getConnection();                      //获得数据库连接
            ps = con.prepareStatement(sql);             //获得预编译语句
            ps.setInt(1, Integer.valueOf(r_id));        //设置 PreparedStatement 的参数
            rs = ps.executeQuery();                     //执行查询
            while(rs.next()){                           //遍历结果集
                String date = rs.getString(1);
                String content = new String(rs.getString(2).getBytes("ISO-8859-1"), CHAR_
                ENCODING);
                String uname = new String(rs.getString(3).getBytes("ISO-8859-1"),CHAR_
                ENCODING);
                String uno = rs.getString(4)+"";
                Comments c = new Comments(date,content,uname,uno);//创建 Comment 对象
                result.add(c);
```

```
22                  }
23          }catch(Exception e){e.printStackTrace();}          //捕获并打印异常
24          finally{
25                  try{if(rs != null){rs.close();rs = null;}      //关闭结果集 ResultSet 对象
26                  }catch(Exception e){e.printStackTrace();}  //捕获并打印异常
27                  try{if(ps != null){ps.close(); ps = null;}     //关闭预编译语句对象
28                  }catch(Exception e){e.printStackTrace();}  //捕获并打印异常
29                  try{if(con != null){con.close();con = null;}   //关闭数据库连接
30                  }catch(Exception e){e.printStackTrace();}  //捕获并打印异常
31          }
32          return result;                                     //返回查询结果
33      }
```

☑ 第 6 行创建了用于查询的 SQL 语句，该 SQL 语句检索 comment 和 user 表，将评论日期、评论内容、评论者昵称及 ID 信息按照一定的条件检索出来，并按照日期时间降序排列。

☑ 第 13 行将 getComments 方法的参数转为 int 类型并设置为预编译语句的参数。第 14 行代码执行查询并将结果赋值给 ResultSet 对象。

☑ 第 15～22 行为读取结果集中的每条记录，并使用从这些记录中提取的内容创建 Comments 对象。Comments 对象封装了与日志评论相关的信息，该类比较简单，代码不予列出。

（3）编辑和删除日志功能的开发

对于 diary.jsp 来说，如果是用户查看自己的日志，即查询日志时提供的用户 ID 与当前 Session 中存放的用户 ID 相同，那么会在显示日志及评论的同时向用户提供编辑和删除的功能。这两个功能的实现比较简单。MyServlet 处理日志编辑动作的代码如下。

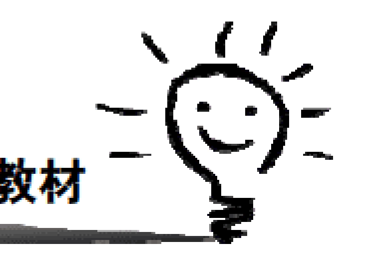
```
1   else if(action.equals("modifyDiary")){                               //action 为修改日志
2               String rid = request.getParameter("r_id");               //读取日志编号参数
3               String rtitle = request.getParameter("r_title");         //读取日志标题参数
4               String rcontent = request.getParameter("r_content");     //读取日志内容参数
5               int result = DBUtil.modifyDiary(rid, rtitle, rcontent);  //调用 modifyDiary 方法
6               request.setAttribute("result", result);                  //设置 request 属性
7               request.getRequestDispatcher("modifyDiary.jsp").forward(request, response);
                                                                         //跳转到 modifyDiary.jsp
8   }
```

☑ 第 2～4 行获取要修改的日志的新标题和内容。

☑ 第 5 行调用了 DBUtil 类的 modifyDiary 静态方法修改日志，并在第 6 行将该方法的返回值设置到 request 的属性中。

☑ 第 7 行跳转到 modifyDiary.jsp，即用户在 diary.jsp 页面单击“编辑”日志的超链接会跳转到 modifyDiary.jsp，在该页面进行日志的修改后提交 action 为 modifyDiary 的请求到 MyServlet。

在上述代码中调用到了 DBUtil 类的 modifyDiary 方法，该方法的功能是更新数据库中指定日志标号的记录，其代码如下。



```
1   public static int modifyDiary(String rid,String rtitle,String rcontent){ //方法：修改指定日志
2           int result = 0;                                              //记录数据库操作结果
3           Connection con = null;                                       //声明数据库连接对象
4           PreparedStatement ps = null;                                 //声明预编译语句对象
5           try{
6               con = getConnection();                                   //获得数据库连接
7               ps = con.prepareStatement("update diary set r_title=?,r_content=?,r_date=now()
                where r_id=?");
8               ps.setString(1, new String(rtitle.getBytes(CHAR_ENCODING),"ISO-8859-1"));
9               ps.setString(2, new String(rcontent.getBytes(CHAR_ENCODING),"ISO-8859-1"));
10              ps.setInt(3, Integer.valueOf(rid));
11              result = ps.executeUpdate();                             //执行更新
12          }catch(Exception e){e.printStackTrace();}                    //捕获并打印异常
13          finally{
14              try{if(ps != null){ps.close(); ps = null;}               //关闭预编译语句对象
15              }catch(Exception e){e.printStackTrace();}                //捕获并打印异常
16              try{if(con != null){con.close();con = null;}             //关闭数据库连接
17              }catch(Exception e){e.printStackTrace();}                //捕获并打印异常
18          }
19          return result; //返回数据库更新结果
20      }
```

☑ 第 3 行和第 4 行声明了数据库连接对象和预编译语句对象，这些对象将会在第 13～18 行的 finally 语句中被关闭。

☑ 第 7～10 行为预编译语句对象设置参数，第 11 行中调用 preparedStatement 的 executeUpdate 方法更新数据库中的记录，第 19 行将返回更新结果。

说明

当用户单击日志页面的“删除”超链接时，会向 MyServlet 发送 action 为“deleteDiary”的请求，MyServlet 会调用 DBUtil 类的 deleteDiary 方法执行删除操作。该业务流程比较简单，由于篇幅有限，在此不再赘述。

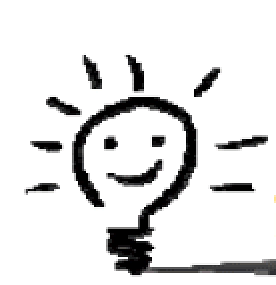
4. 查看和管理相册模块的实现

除了日志的查看和管理，本系统还为用户提供了图片的分享平台，这些服务通过相册的查看和管理模块来实现，下面就对这个功能模块进行简单的介绍。

（1）查看相册功能的开发

用户登录后，在页面中单击“查看相册”会向 MyServlet 发出 action 为“seeAlbum”的请求，MyServlet 处理该请求的代码如下。

```
else if(action.equals("seeAlbum")){                                  //action 为查看相册
        HttpSession session = request.getSession();                   //获得 Session 对象
        User user = (User)session.getAttribute("user");              //读取 Session 中的 User 对象
```



```
        if(user!=null){                                    //判断 User 对象是否为空
            request.setAttribute("u_no", user.u_no);       //设置返回属性
        }
        request.getRequestDispatcher("album.jsp").forward(request, response);
                                                           //跳转到 album.jsp
    }
```



说明

上述代码中首先获得 Session 对象，然后读取 Session 对象中的 User 对象，如果该对象不为空，则将 User 对象设置为 request 的属性并跳转到 album.jsp。

当 MyServlet 将请求跳转到 album.jsp 之后，album.jsp 通过调用 DBUtil 类的方法获取指定用户的相册列表并采用特定的方式呈现到网页上。

获取相册列表有两个方法：getAlbumList 和 getAlbumListByAccess，这两个方法的区别是 getAlbumList 方法获得指定用户的所有相册，而 getAlbumListByAccess 方法将根据相册的访问权限和请求相册的用户 ID 决定返回哪些相册。首先来看 getAlbumListByAccess 方法的代码。

```
public static ArrayList<String []> getAlbumListByAccess(String uno,String visitor){
                                                           //方法：根据权限获取相册
    ArrayList<String []> result = new ArrayList<String []>();  //创建存放相册信息的 ArrayList
    Connection con = null;                                 //声明 Connection 对象
    PreparedStatement ps = null;                           //声明 PreparedStatement 对象
    ResultSet rs = null;                                   //声明 ResultSet 对象
    try{
        con = getConnection();                     //获得数据库连接
        if(checkIfMyFriend(uno, visitor)){         //检查访问者和被访问者是否为好友
        ps = con.prepareStatement("select x_id,x_name,x_access from album
        where u_no=?");                            //创建 SQL 语句
        }
        else{
            ps = con.prepareStatement("select x_id,x_name from album
                                  where u_no=? and x_access=0");//创建 SQL 语句
        }
        ps.setInt(1, Integer.calueOf(uno));
        rs = ps.executeQuery();                    //执行查询
        while(rs.next()){                          //如果结果集中有数据
            String xid = rs.getInt[1]+ "";
            String xname = new String(rs.getString(2).getBytes("ISO-8859-1"),CHAR_ENCODING);
            String [] sa = new String[]{xid,xname};
            result.add(sa);                        //将相册信息添加到 ArrayList 中
        }
    }catch(Exception e){e.printStackTrace();}      //捕获并打印异常
    finally{
        try{if(ps != null){ps.close();ps = null;}  //关闭预编译语句
```

```
27                }catch(Exception e){e.printStackTrace();}     //捕获并打印异常
28                try{if(con != null){con.close();con = null;}  //关闭数据库连接
29                }catch(Exception e){e.printStackTrace();}     //捕获并打印异常
30            }
31            return result;                                    //返回查询的相册信息
32        }
```

☑ 第 2～5 行声明了执行数据库时用到的数据库连接等对象，同时声明了用于存放相册信息的 ArrayList 列表。

☑ 第 8 行调用了 DBUtil 类的 checkIfMyFriend 方法，该方法的功能是检查访问者是否为被访问者的好友并返回相应的 boolean 值。第 9 行和第 13 行根据 checkIfMyFriend 的返回值创建不同权限的 SQL 查询语句。

☑ 第 17 行执行创建好的 SQL 查询语句，并在第 18～23 行将结果集中的数据添加到 ArrayList 中，最终将该 ArrayList 对象返回。

（2）查看相册照片功能的实现

当用户在页面中选择了一个相册并单击“确定”按钮时，album.jsp 通过调用 DBUtil 类的 getPhotoInfoByAlbum 获得指定相册的照片列表，该方法代码如下。

```
public static ArrayList<PhotoInfo> getPhotoInfoByAlbum(String xid,int pageNo,int span){
        ArrayList<PhotoInfo> result = new ArrayList<PhotoInfo>();
                                                //声明用于返回的相片信息列表
        Connection con = null;                  //声明数据库连接对象
        PreparedStatement ps = null;            //声明预编译语句
        ResultSet rs = null;                    //声明 ResultSet 对象
        int start = span*(pageNo-1);            //计算起始位置
        try{
            con = getConnection();              //获得连接
            ps = con.prepareStatement("select p_id,p_name,p_des,x_id from
            photo" +" where x_id=? order by p_id limit "+start+","+span);  //创建语句
            ps.setInt(1, Integer.valueOf(xid)); //设置预编译语句的参数
            rs = ps.executeQuery();
            while(rs.next()){                   //遍历结果集
                String p_id = rs.getInt(1)+"";
                String p_name = new String(rs.getString(2).getBytes("ISO-8859-1"),CHAR_
                ENCODING);//相片名称
                String p_des = new String(rs.getString(3).getBytes("ISO-8859-1"),CHAR_
                ENCODING);//相片描述
                String x_id = rs.getInt(4)+"";
                PhotoInfo p = new PhotoInfo(p_id, p_name, p_des, x_id);//创建 PhotoInfo 对象
                result.add(p);
            }
        }catch(Exception e){e.printStackTrace();}    //捕获并打印异常
        finally{
            try{if(rs != null){rs.close();rs = null;}  //关闭结果集 ResultSet 对象
        }catch(Exception e){e.printStackTrace();}    //捕获并打印异常
```

```
25                  try{if(ps != null){ps.close(); ps = null;}        //关闭预编译语句对象
26                  }catch(Exception e){e.printStackTrace();}        //捕获并打印异常
27                  try{if(con != null){con.close();con = null;}      //关闭数据库连接
28                  }catch(Exception e){e.printStackTrace();}        //捕获并打印异常
29              }
30              return result;                                        //返回相片信息列表
31      }
```

- ☑ getPhotoInfoByAlbum 方法可以支持分页，因此该方法除接收相册编号作为参数之外，还接收代表页号的 pageNo 参数和代表每页显示照片数的 span 参数。如果希望每页只显示一张图片，那么设置 span 为 1 即可。
- ☑ 第 2～5 行声明了操作数据库所需要的数据库连接等对象，以及用于返回的存放照片信息的 ArrayList 列表。
- ☑ 第 8～12 行分别进行了连接数据库、创建预编译语句及执行查询的操作，查询结果存放在 ResultSet 对象中。
- ☑ 第 13～20 行遍历结果集并根据查询到的字段值创建 PhotoInfo 对象添加到 ArrayList 列表中。PhotoInfo 类中封装了包括照片编号、相片名称、相片描述及所属相册的信息，该类的实现比较简单，由于篇幅所限，不再详述其代码。

（3）显示照片功能的实现

在 album.jsp 页面中通过调用 DBUtil 的 getPhotoInfoByAlbum 方法获得了指定相册的照片信息，这些文本信息将按照特定的格式显示在页面中，而照片本身是通过 photo.jsp 来显示的。该 JPS 文件的代码如下。

```
<%@ page language="java" contentType="text/html; charset=GBK"
    pageEncoding="GBK"import="wpf.DBUtil,wpf.ConstantUtil, java.util.*,java.sql.*,java.io.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<LINK href="global.css" type="text/css" rel="stylesheet"/>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
<title></title>
</head>
<body>
<%
        request.setCharacterEncoding(ConstantUtil.CHAR_ENCODING);    //设置编码格式
        String pid = (String)request.getParameter("pid");            //读取照片 ID
        Blob blob = DBUtil.getPhotoBlob(pid);               //调用 DBUtil 类的方法获得图片 Blob
        if(blob == null){                                   //获取 Blob 图片失败
            response.sendRedirect("img/no_image.jpg");      //显示默认的图片
        }
        else{                                               //成功获取数据
            long size = blob.length();                      //获取长度
```

```
21                byte [] bytes = blob.getBytes(1,(int)size);           //获取字节数组
22                response.setContentType("image/jpeg");
23                OutputStream outs = response.getOutputStream();  //获得输出流
24                outs.write(bytes);                                    //将图片数据
25                outs.flush();                                         //清空缓存
26                out.clear();                                          //清除 out
27                out = pageContext.pushBody();                         //返回 PageContent 对象
28         }
29    %>
30    </body></html>
```

☑ 第 13 行设置 request 的编码格式，这样做是为了防止出现乱码。

☑ 第 14 行读取代表照片 ID 的参数 pid，并在代码第 15 行调用 DBUtil 的 getPhotoBlob 方法获得图片的 Blob 对象。

☑ 第 16～18 行代码首先判断 getPhotoBlob 方法返回的 Blob 对象是否为 null，如果为 null，则显示系统默认的图片。

☑ 第 19～28 行为当 getPhotoBlob 的返回值不是 null 时执行的代码，首先获取 Blob 对象的长度，然后将 Blob 对象中的数据读取到字节数组中。代码第 22 行设置了 photo.jsp 的 ContentType 为“image/jpeg”。代码第 24 行将图片的字节数组发送到浏览器客户端。

photo.jsp 的使用也比较简单，只需要将 HTML 的<img>标记中的“scr”属性设置为 photo.jsp 并传入代表照片编号的参数即可。album.jsp 中调用 photo.jsp 的代码片段如下。

```
<img src="photo.jsp>pid=<%p_id %>" width="400" height="400"></img>
```



说明

> 其中 pi 为 PhotoInfo 类的对象，将该对象中的 p_id 属性作为参数传递给 photo.jsp。显示用户头像的功能实现和显示照片类似，是通过 head.jsp 实现的，该 JSP 文件的代码与 photo.jsp 基本类似。

在前面的内容中对微博随身系统的 Web 端的主要功能模块的实现已经进行了介绍，由于此处的重点是讲解 Android 的开发，对 Web 端只是进行简单的介绍。下面将会详细地介绍微博随身 Android 端的设计与实现。

小知识八：Web 前端开发规范

1. 规范目的

为提高团队协作效率，便于后台人员添加功能及前端后期优化维护，输出高质量的文档，特制订此文档。本规范文档一经确认，前端开发人员必须按本文档规范进行前台页面开发。本文档如有不对或者不合适的地方请及时提出，经讨论决定后方可更改。

2. 基本准则

符合 Web 标准，语义化 html，结构表现行为分离，兼容性优良。页面性能方面，代码



要求简洁明了、有序，尽可能地减小服务器负载，保证最快的解析速度。

3. 文件规范

（1）html，css，js，images 文件均归档至<系统开发规范>约定的目录中。

（2）html 文件命名：英文命名，后缀.htm. 同时将对应界面稿放于同目录中，若界面稿命名为中文，请重命名与 html 文件同名，以方便后端添加功能时查找对应页面。

（3）css 文件命名：英文命名，后缀.css 共用 base.css，首页 index.css，其他页面依实际模块需求命名。

（4）js 文件命名：英文命名，后缀.js 共用 common.js，其他依实际模块需求命名。

4. html 书写规范

（1）文档类型声明及编码：统一为 html5 声明类型<!DOCTYPE html>；编码统一为<meta charset="gbk" />，书写时利用 IDE 实现层次分明的缩进。

（2）非特殊情况下样式文件必须外链至<head></head>之间；非特殊情况下 JavaScript 文件必须外链至页面底部。

（3）引入样式文件或 JavaScript 文件时，需略去默认类型声明，写法如下：

```
Example Source Code [www.52css.com]
<link rel="stylesheet" href="..." />
<style>...</style>
<script src="..."></script>
```

（4）引入 JS 库文件，文件名需包含库名称、版本号及是否为压缩版，比如 jquery-1.4.1.min.js；引入插件，文件名格式为库名称+插件名称，比如 jQuery.cookie.js。

（5）所有编码均遵循 xhtml 标准，标签&属性&属性命名必须由小写字母及下划线数字组成，且所有标签必须闭合，包括 br（
），hr（<hr/>）等；属性值必须用双引号包括。

（6）充分利用无兼容性问题的 html 自身标签，比如 span、em、strong、optgroup、label 等；需要为 html 元素添加自定义属性的时候，首先要考虑有没有默认的已有的合适标签设置，如果没有，可以使用以"data-"为前缀来添加自定义属性，避免使用"data:"等其他命名方式。

（7）语义化 html，如标题根据重要性用 h*（同一页面只能有一个 h1），段落标记用 p，列表用 ul，内联元素中不可嵌套块级元素。

（8）尽可能减少 div 嵌套，如<div class="box"><div class="welcome">欢迎访问 XXX，您的用户名是<div class="name">用户名</div></div></div>，完全可以用以下代码替代：<div class="box"><p>欢迎访问 XXX，您的用户名是用户名</p></div>。

（9）书写链接地址时，必须避免重定向，例如，href=http://itaolun.com/，即在 URL 地址后面加上“/”。

（10）在页面中尽量避免使用 style 属性，即 style="…"。

（11）必须为含有描述性表单元素（input，textarea）添加 label。

（12）能以背景形式呈现的图片，尽量写入 css 样式中。



（13）重要图片必须加上 alt 属性；给重要的元素和截断的元素加上 title。

（14）给区块代码及重要功能（比如循环）加上注释，方便后台添加功能。

（15）特殊符号使用：尽可能使用代码替代：例如<(<) & >(>) &空格( ) & »(»)等。

（16）书写页面过程中，请考虑向后扩展性。

（17）class & id 参见 CSS 书写规范。

5. CSS 书写规范

（1）编码统一为 UTF-8。

（2）协作开发及分工：i 会根据各个模块，同时根据页面相似程序，事先写好大体框架文件，分配给前端人员，实现内部结构&表现&行为；共用 CSS 文件 base.css 由 i 书写，协作开发过程中，每个页面务必都要引入，此文件包含 reset 及头部底部样式，此文件不可随意修改。

（3）class 与 id 的使用：id 是唯一的并是父级的，class 是可以重复的并是子级的，所以 id 仅使用在大的模块上，class 可用在重复使用率高及子级中；id 原则上都是由分发框架文件时命名的，为 JavaScript 预留的除外。

（4）为 JavaScript 预留的命名，应以 js_起始，比如：js_hide, js_show。

（5）class 与 id 命名：大的框架命名，比如 header/footer/wrapper/left/right 之类由 i 统一命名。其他样式名称由小写英文&数字&_来组合命名，如 i_comment, fontred, width200；避免使用中文拼音，尽量使用简易的单词组合；总之，命名要语义化、简明化。

（6）规避 class 与 id 命名（此条重要，若有不明白请及时与 i 沟通）。

① 通过从属写法规避。

② 取父级元素 id/class 命名部分命名。

③ 重复使用率高的命名，请以自己代号加下划线起始，比如 i_clear。

④ ①②两条，适用于已建好框架的页面。

（7）css 属性书写顺序，建议遵循：布局定位属性→自身属性→文本属性→其他属性。此条可根据自身习惯书写，但尽量保证同类属性写在一起。

属性列举：

布局定位属性主要包括：display & list-style & position（相应的 top, right, bottom, left）& float & clear & visibility & overflow。

自身属性主要包括：width & height & margin & padding & border & background。

文本属性主要包括：color & font & text-decoration & text-align & vertical-align & white-space &。

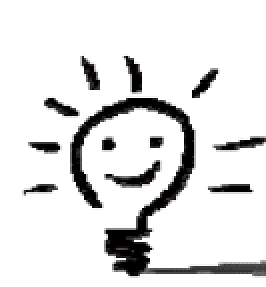
其他属性：& content。

（8）书写代码前，要考虑并提高样式重复使用率。

（9）充分利用 html 自身属性及样式继承原理减少代码量。

（10）样式表中中文字体名，请务必转码成 unicode 码，以避免编码错误时乱码。

（11）背景图片请尽可能使用 Sprite 技术，减小 HTTP 请求，考虑到多人协作开发，sprite 按模块制作。



（12）使用 table 标签时（尽量避免使用 table 标签），不要用 width/ height/cellspacing/ cellpadding 等 table 属性直接定义表现，应尽可能地利用 table 自身私有属性分离结构与表现。

（13）杜绝使用<meta http-equiv="X-UA-Compatible" content="IE=7" /> 兼容 IE8。

（14）用 png 图片做图片时，要求图片格式为 png-8 格式，若 png-8 实在影响图片质量或其中有半透明效果，为 ie6 单独定义背景。

（15）避免兼容性属性的使用，比如 text-shadow || css3 的相关属性。

（16）减少使用影响性能的属性，比如 position:absolute || float。

（17）必须为大区块样式添加注释，小区块适量注释。

（18）代码缩进与格式：建议单行书写，可根据自身习惯，后期优化 i 会统一处理。

6. JavaScript 书写规范

（1）文件编码统一为 UTF-8，书写过程中，每行代码结束必须有分号（;），原则上所有功能均根据 XXX 项目需求原生开发，以避免网上下载下来的代码造成的代码污染（沉冗代码、与现有代码冲突或其他情况）。

（2）库引入：原则上仅引入 jQuery 库，若需引入第三方库，需与团队其他人员讨论决定。

（3）变量命名：驼峰式命名，原生 JavaScript 变量要求是纯英文字母，首字母必须小写，如 iTaoLun。

（4）类命名：首字母大写，驼峰式命名，如 ITaoLun。

（5）函数命名：首字母小写驼峰式命名，如 iTaoLun()。

（6）命名语义化，尽可能利用英文单词或其缩写。

（7）尽量避免使用存在兼容性及消耗资源的方法或属性，比如 eval() & innerText。

（8）后期优化中，JavaScript 非注释类中文字符需转换成 unicode 编码再使用，以避免编码错误时乱码显示。

（9）代码结构明了，加适量注释，以提高函数重用率。

（10）注重与 html 分离，减小 reflow，注重性能。

7. 图片规范

（1）所有页面元素类图片均放入 img 文件夹，测试用图片放于 img/demoimg 文件夹中。

（2）图片格式仅限于 gif、png、jpg。

（3）命名全部用小写英文字母、数字、_的组合，其中不得包含汉字、空格、特殊字符；尽量用易懂的词汇，便于团队其他成员理解；另外，命名分头尾两部分，用下划线隔开，比如 ad_left01.gif、btn_submit.gif。

（4）在保证视觉效果的情况下选择最小的图片格式与图片质量，以减少加载时间。

（5）尽量避免使用半透明的 png 图片（若使用，请参考 CSS 规范相关说明）。

（6）运用 css sprite 技术集中小的背景图或图标，减小页面 HTTP 请求，但注意，务必在对应的 sprite psd 源图中画参考线，并保存至 img 目录下。



8. 注释规范

（1）HTML 注释：注释格式<!--这儿是注释-->，“--”只能在注释的始末位置，不可置入注释文字区域。

（2）CSS 注释：注释格式 /*这里是注释*/。

（3）JavaScript 注释，单行注释使用：//这里是单行注释，多行注释使用：/*这里有多行注释 */。

9. 开发及测试工具约定

建议使用 Aptana、Dw、Vim，亦可根据自己喜好选择，但须遵循以下原则：

（1）不可利用 IDE 的视图模式“画”代码。

（2）不可利用 IDE 生成相关功能代码，比如 Dw 内置的一些功能 js。

（3）编码必须格式化，比如缩进。

测试工具：前期开发仅测试 FireFox & IE6 & IE7 & IE8，后期优化时加入 Opera & Chrome & Safari。

建议测试顺序：FireFox→IE7→IE8→IE6→Opera→Chrome→Safari，建议安装 firebug 及 IE Tab Plus 插件。

10. 其他规范

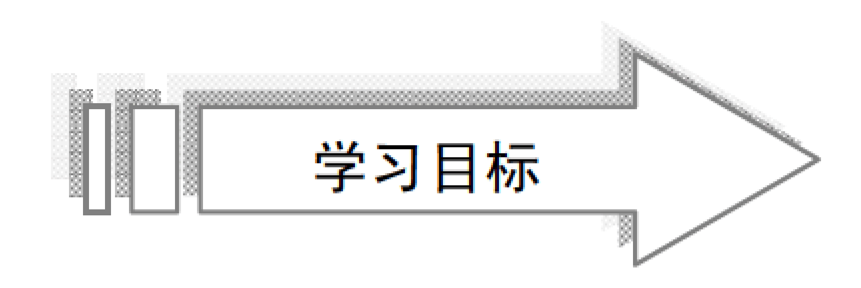
（1）开发过程中严格按分工完成页面，以提高 CSS 重用率，避免重复开发。

（2）减小沉冗代码，书写所有人都可以看得懂的代码，简洁易懂是一种美德，要为用户着想，为服务器着想。

项目七
Activity和游戏工具类的开发



前面已经介绍了游戏的整体框架，接下来将正式进入游戏的开发，本项目介绍的是主控制器 Activity 以及游戏工具相关类。



- 学习搭建 HDZGActivity 框架，进行界面的切换与控制
- 学习公式封装类 GameFormula，提供一系列游戏中的计算公式
- 学习常用工具类 ConstantUtil，封装游戏中的所有常量

任务 13　HDZGActivity 类

【任务情境】

HDZGActivity 的工作主要是进行界面的切换与控制，根据收到的 Handler 消息做出相应的操作。

【相关知识】

具体开发步骤如下。

（1）搭建 HDZGActivity 的框架，其代码如下。

```
package wyf.ytl;
import android.app.Activity;                    //引入相关类
import android.os.Bundle;                       //引入相关类
import android.os.Handler;                      //引入相关类
import android.os.Looper;                       //引入相关类
import android.os.Message;                      //引入相关类
import android.view.KeyEvent;                   //引入相关类
import android.view.View;                       //引入相关类
import android.view.Window;                     //引入相关类
import android.view.WindowManager;              //引入相关类
public class HDZGActivity extends Activity {
//对游戏中的所有界面进行声明
    LoadingView loadingView;                    //进度条界面的引用
    MenuView menuView;                          //开始游戏的主菜单界面
    GameView gameView;                          //游戏主界面
    HelpView helpView;                          //帮助界面
    SoundView soundView;                        //刚开始时选择是否开启声音的界面
    SoundManageView soundManageView;            //声音设置界面
    ScreenRollView screenRoll;                  //竹筒滚屏界面
    AboutView aboutView;                        //关于界面
/*4 种声音是否播放的标志位，包括背景音、开场音、战斗音以及环境音，每次需要播放声音时只需
判断相应的标志位即可*/
    boolean isBackSound = true;                 //是否有背景声音
    boolean isStartSound = true;                //开场声音
    boolean isBattleSound = true;               //战斗声音
    boolean isEnvironmentSound = true;          //环境声音
    View currentView = null;                    //当前的 View
/*Handler 消息接收对象，在 handleMessage 方法中，根据得到消息类型的不同执行不同的操作或调
用不同的方法*/
Handler myHandler = new Handler(){              //用来更新 UI 线程中的控件
```

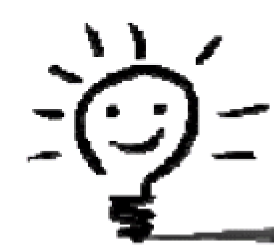


```
                public void handleMessage(Message msg) {    //接收 Handler 消息
                    switch(msg.what){//分支判断
                    …//该处省略了根据接收到消息的不同做出相应操作的代码，后面将给出
                    }
                }
        }
        public void onCreate(Bundle savedInstanceState) {            //重写的 onCreate 方法
                super.onCreate(savedInstanceState);
                requestWindowFeature(Window.FEATURE_NO_TITLE);  //全屏
                getWindow().setFlags (Window Manager.LayoutParams.FLAG_FULLSCREEN, WindowManager.
                Layout Params.FLAG_FULLSCREEN);
                soundView = new Sound View(this);                   //初始化询问声音开启的界面
                this.setContentView(soundView);                     //先切到简单的声音开关界面
        /*因为所有的图片只能加载一次，再次使用时无须再次加载，所以各个界面对象中的图片资源与
resources 全部为静态的，在初始化 Activity 时便可对各个界面的图片进行初始化，如本部分代码
                GameData.resources=this.getResources();             //为 GamData 的静态变量赋值
                GameData.initMapImage();                            //初始化地图图片
                GameData.initMapData();                             //初始化地图数据
                GameData2.resources=this.getResources();            //为 GamData2 的静态变量赋值
                GameData2.initBitmap();                             //初始化地图图片
                GameData2.initMapData();                            //初始化地图数据
                GameView.initBitmap(this.getResources());           //初始化 GameView 的图片
                BattleField.initBitmap(this.getResources());        //初始化 BattleField 的图片
                ManPanelView.initBitmap(this.getResources());
                WuJiangView.initBitmap(this.getResources());
                UseSkillView.initBitmap(this.getResources());
                CityManageView.initBitmap(this.getResources());
                SelectGeneral.initBitmap(this.getResources());      //初始化 SelectGeneral 的图片
                TianXiaView.initBitmap(this.getResources());        //初始化 TianXiaView 界面的图片
                }
        /*各个方法的作用是将当前的用户界面切换到相应界面*/
        public void toLoadingView(int toViewID){                    //切换到加载界面
                  loadingView = new LoadingView(this,toViewID);     //初始化进度条
                  this.currentView = loadingView;
                  this.setContentView(loadingView);                 //切换到进度条 View
        }
            public void toMenuView(){                               //切换到开始游戏的主菜单
                if(this.menuView != null){                          //当 menuView 不为空时切屏
                    this.currentView = menuView;
                    this.setContentView(menuView);
                }else{                                              //当为空时打印并退出
                    System.exit(0);                                 //退出
                }
            }
        public void toGameView(){                                   //切换到游戏主界面
                if(this.gameView != null){                          //当 gameView 不为空时
```

```
                this.currentView = gameView;
                this.setContentView(gameView);
            }else{                                        //当为空时打印并退出
                System.exit(0);
            }
    }
    /*重写的按键监听方法，当有按键被按下时，只需调用当前显示界面的 onKeyDown 方法，无须做出
其他操作*/
    public boolean onKeyDown(int keyCode, KeyEvent event) {          //重写的按键监听方法
        if(currentView != null){
                currentView.onKeyDown(keyCode, event);       //调用各个 View 相应的方法即可
        }
        return super.onKeyDown(keyCode, event);
        }
    }
    */
```

（2）对上面代码中方法 handleMessage 进行完善，其代码如下。

```
    switch(msg.what){                                          //分支判断
        /*收到加载界面发来的类型为 1 的消息，说明菜单界面加载完毕，需要将当前用户界面切换到菜
单界面*/
    case 1:                                                    //收到 LoadingView 的消息，切到 MenuView
            if(loadingView != null){
                loadingView = null;                            //释放 LoadingView
            }
            toMenuView();                                      //切换到开始游戏时的主菜单
            break;
            .../*省略了收到其他界面消息后切换到相应界面的处理代码，其处理方式基本相同，都是先
判断消息的类型，然后将用户界面切换到相应的用户界面*/
        /*收到风云再起事件的相应代码，先判断存档文件是否存在，因为只有存在时才可以加载存档，
然后初始化游戏界面 GameView 并加载存档信息到 gameView 中。将耗时的初始化工作放到线程中是为了
保证前台用户界面的流畅*/
    case 99:                                                   //收到风云再起的事件响应
            if(loadingView != null){
                loadingView = null;                            //释放 LoadingView
            }
            currentView = null;                                //将表示当前用户界面置空
            if(SerializableGame.check(HDZGActivity.this)){     //检测之前是否有过存档
                toLoadingView(101);                            //切换到加载界面
                new Thread(){                                  //创建线程
                    public void run(){
                        try{
                            Thread.sleep(2000);                //睡眠 2000 毫秒
                        }catch(Exception e){                   //捕获异常
                            e.printStackTrace();               //打印异常信息
```



```
                                        }
                                        Looper.prepare();
                                        if(gameView==null){              //当 gameView 为空时
                                            gameView = new GameView(HDZGActivity.this);//初始化 GameView
                                        }
SerializableGame.loadingGameStatus(gameView);                            //加载存档的信息
                                    }
                            }.start();                                   //启动线程
                    }
                    break;
*/
case 100:
if(gameView != null){                                                    //当 gameView 为空时
            HDZGActivity.this.setContentView(gameView);                  //初始化游戏界面 gameView
        }
        break;
```

任务 14　公式封装类 GameFormula

【任务情境】

GameFormula 类提供了一系列的静态方法，主要是游戏过程中所用到的所有开发公式，例如，攻城输赢的判断、伐木所得金额的多少、体力增加的多少等。

【相关知识】

具体代码如下。

```
package wyf.ytl;
/*该类提供一系列的静态方法，主要是一些公式的计算，如伐木所得、攻城输赢计算、体力增加等*/
public class GameFormula{
    public static final int BASIC_STRENGTH_INCREMENT = 18;    //英雄的体力基础增加值
    public static final int EXTRA_STRENGTH_INCREMENT = 2;     //英雄的体力额外增加值和等级有关
    public static final int FOOD_DECREASE_EACH = 8;           //每个人消耗的粮食
    /*计算英雄应该增加体力多少的方法，增加的体力为英雄基础增加值加上英雄等级与额外增加值的
乘积*/
    //计算并返回英雄应该增加的体力
    public static int getHeroStrengthIncrement(Hero hero){    //根据英雄的现状计算体力该增加多少
        int level = hero.level;
        int increment = BASIC_STRENGTH_INCREMENT+level*EXTRA_STRENGTH_INCREMENT;
        if(hero.strength+increment >hero.maxStrength){
            increment = hero.maxStrength-hero.strength;
        }
```

```
        return increment;//返回计算结果
    }
/*计算将领的体力增加值，该算法比较简单，只将将领的基础增加值返回*/
计算并返回将领应该增加的体力，基本算法就是体力越低，恢复越快，等级越高，恢复越慢，但暂时先不这样计算*/
    public static int getGeneralStrengthIncrement(General general){
        int level = general.getLevel();                   //获得将军等级
        int strength = general.getStrength();             //获得将军体力
        int increment = BASIC_STRENGTH_INCREMENT; //计算出增量
        if(strength + increment > general.maxStrength){   //如果不用加那么多就已经完全恢复
            increment = general.maxStrength - strength;   //恢复满
        }
        return increment;
    }
/*计算粮食的衰减数，为了演示方便，只是返回每个消耗的粮食与英雄等级的和，即英雄等级越高消耗的粮食也就会越多*/
//计算并返回粮草的衰减数
    public static int getFoodDecreaseEach(Hero hero){
        int level = hero.level;                           //英雄的等级
        return FOOD_DECREASE_EACH+level;
    }
/*计算技能的收益值，例如，收麦子、收稻田、捕鱼等，此处将这些算法简化并封装到同一方法中。在真实的游戏开发中，应该将其分为多个方法且应用更复杂、更合理的算法*/
//计算并返回技能的收益值，如该获得多少麦子等
    public static int getSkillEearning(int proficiency,int basicEarning){
        return basicEarning*(proficiency);
    }
/*得到指定城池的总防御力，该防御力是根据镇守武将的信息以及城池中箭垛等防御设施的多少计算。而第 41～47 行方法计算的是指定城池的总攻击力*/
//计算并返回一座城池的防御力
    public static int getCityDefence(CityDrawable city){
        int defend =0;                                              //定义临时变量
        General g = city.guardGeneral.get(0);                       //得到城池中的主将
        float gDefend = (g.defend*0.8f+g.intelligence*0.2f)/100.0f; //计算
        defend = (int)(city.baseDefend*city.army*(1+gDefend)) +
city.warTower*100;
        return defend;                                              //返回计算结果
    }
//计算并返回一座城池的攻击力
public static int getCityAttack(CityDrawable city){
        int attack = 0;                                             //攻击力
        General g = city.getGuardGeneral().get(0);                  //获得守将对象
        float gAttack = (g.power*0.7f+g.agility*0.2f+g.intelligence*0.1f)/100.0f;
        attack = (int)(city.baseAttack*city.getArmy()*(1+gAttack)) + city.warTank*100;
        return (int)attack;                                         //返回攻击力
    }
```

```
/*英雄攻击力的计算方法，是根据其武力、智力等信息综合计算的，第 57～62 行则根据英雄的信息计算基础防御力*/
    //计算并返回英雄的攻击力
    public static int getHeroAttack(Hero hero,General general){
        int attack = 0;                                          //攻击力
        float gAttack = (general.power*0.7f+general.agility*0.2f+general.intelligence*0.1f)/100.0f;
                                                                 //英雄的攻击加成
        attack = (int)((hero.basicAttack*hero.armyWithMe)*(1+gAttack));
        return attack;
    }
   */
    //计算并返回英雄的防御力
    public static int getHeroDefence(Hero hero,General general){
        int defence = 0;                                         //防御力
        float gDefend = (general.defend*0.8f+general.intelligence*0.2f)/100.0f;   //英雄的防御加成
        defence = (int)((hero.basicDefend*hero.armyWithMe)*(1+gDefend));
        return defence;                                          //返回防御力
    }
}
```

任务 15　常量工具类 ConstantUtil

【任务情境】

前面已经对公式计算类进行了介绍，接下来将介绍另一个工具类——常量工具类，该类中封装了游戏中所用到的所有常量，该常量封装到一个常量类中的好处是便于管理和维护。

【相关知识】

其代码如下。

```
package wyf.ytl;
import java.util.ArrayList;
import java.util.Collections;
public class ConstantUtil {//常量类
/*对各个类中所用到的常量进行定义*/
    //MenuView 界面中用到的常量
    public static final int MENU_VIEW_SLEEP_SPAN = 200;//MenuView 界面刷帧线程睡眠时间
    …//省略了 MenuView 界面中部分常量的声明
    public static final int PICTUREHEIGHT = 480;
     //LoadingView 界面中用到的常量
public static final int LOADING_VIEW_WORD_SIZE = 12;//进度条界面中“加载中”3 个字的大小
   …//省略了 LoadingView 界面中部分常量的声明
```

```
        public static final int LOADING_VIEW_SLEEP_SPAN = 400;//进度条界面刷帧线程睡眠时间
        //GameView 界面中用到的常量
        public static final int GAME_VIEW_SLEEP_SPAN = 100;//GameView 界面刷帧线程睡眠时间
       …//省略了 GameView 界面中部分常量的声明
        public static final int STRENGTH_COST_DECREMENT = 2;//每次技能升级减小的体力消耗值
        //英雄类中用到的常量
        public static final int HERO_ANIMATION_SEGMENTS = 8;//英雄总共的动画段个数
       …//省略了英雄类中部分常量的声明
        public static final int RIGHT = 6;//英雄的移动方向向右，也代表相应动画段
    //技能类中用到的常量
    public static final int LUMBER = 0;//代表伐木技能，作为技能 HashMap 的键
    …//省略了技能类中部分常量的声明
        public static final int WU_ZHONG_SHENG_YOU = 6;//代表无中生有
    /*初始化英雄的所有官衔，在游戏过程中需要提升官衔时直接从该数组中提取即可*/
        public static final String[] HERO_TITLE = new String[]{//英雄的官衔"公爵", "男爵"
            …//省略了部分官衔
    };
    /*存储了将领的所有职位，当任免将领时，同样从本数组中取数据*/
    public static final String[] GENERAL_TITLE = new String[]{//将领的职位"总兵","护国将军"
            …//该处省略了部分职位
       };
        public static final String[] RESEARCH_PROJECT = new String[]{//科研的项目"战车", "箭垛"
    };
    /*定义了可科研的项目*/
    //敌方城池中的将领
    public static final ArrayList<General> ENEMY_GENERAL = new ArrayList<General>();
    *初始化敌方城池中用到的所有将领，当初始化敌方城池时，会从中抽取相应的将领到敌方城池中
    static{
    ENEMY_GENERAL.add(new General("赵奢", 5, 94, 80, 69, 69, 78, 80, 8));
    …//省略了部分将军的初始化代码
    ENEMY_GENERAL.add(new General("李斯", 1, 56, 52, 62, 59, 68, 68, 2));
    }
    //所有的城市信息
    public static final ArrayList<CityInfo> CITY_INFO = new ArrayList< CityInfo>();
    /*初始化地图中所有的城池信息，将其存储到 ArrayList 容器中，在初始化时使用了工具类中的 shuffle
方法将所有将领打乱顺序，使得每次加载游戏时敌方城池有一定的随机性，提高了游戏的可玩性*/
    static{
    Collection.shuffle(ENEMY_GENERAL);//打乱顺序
    CITY_INFO.add(new CityInfo("洛阳", 0, 5000, 5000, 2, 22, 15, 80000, 5, 8, ENEMY_GENERAL.get(0), 1));
    …//省略了部分城池的初始化代码
    CITY_INFO.add(new CityInfo("宁波", 7, 6000, 6000, 5, 22, 15, 50000, 7, 12, ENEMY_GENERAL.
get(21), 22));
    }
    public static final String[] COUNTRY_NAME = new String[]{//所有的国家"周国", "秦国", "自己"
       …//省略了部分国家的名称
    };
```



```
    */
    ENEMY_GENERAL.add(new General("李斯", 1, 56, 52, 62, 59, 68, 68, 2));
    }
    //所有的城市信息
    /*定义游戏中的所有国家，城池所属的国家都是该数组中的一种。在代码的第 62～64 行定义了一些
共用的常量，如屏幕的宽度、高度等*/
    public static final int SCREEN_HEIGHT = 480;      //屏幕宽度
    …//省略了一部分其他常量的声明
    public static final int DIGIT_SPAN = 10;           //两个数码管左边沿的距离
    }
```

【项目小结】

从本项目开始正式进入游戏的开发阶段，介绍了主控制器 Activity 以及游戏工具相关类的开发，具体包括 HDZGActivity 类的开发、公式封装类 GameFormula 类的开发和常量工具类 ConstantUtil 类的开发，读者要仔细阅读本项目中的代码，此方法可用于同类其他游戏的开发过程。

📖 小知识九：Activity 组件详解

1. Activity 的生命周期

和 J2ME 的 MIDlet 一样，在 Android 中，Activity 的生命周期交给系统统一管理。与 MIDlet 不同的是安装在 Android 中的所有的 Activity 都是平等的。

2. Activity 的状态及状态间的转换

在 Android 中，Activity 拥有四种基本状态：

（1）Active/Runing 是一个新 Activity 启动入栈后，它在屏幕最前端，处于栈的最顶端，此时它处于可见并可与用户交互的激活状态。

（2）Paused 是 Activity 被另一个透明或者 Dialog 样式的 Activity 覆盖时的状态。此时它依然与窗口管理器保持连接，系统继续维护其内部状态，所以它仍然可见，但已经失去了焦点，故不可与用户交互。

（3）Stoped 是 Activity 被另外一个 Activity 覆盖、失去焦点并不可见时处于的状态。

（4）Killed Activity 是被系统杀死回收或者没有被启动时所处于的状态。

当一个 Activity 实例被创建、销毁或者启动另外一个 Activity 时，它在这四种状态之间会进行转换，这种转换的发生依赖于用户程序的动作。

3. Activity 栈

Android 是通过一种 Activity 栈的方式来管理 Activity 的，一个 Activity 实例状态决定它在栈中的位置。处于前台的 Activity 总是在栈的顶端，当前台的 Activity 因为异常或其他原因被销毁时，处于栈第二层的 Activity 将被激活，上浮到栈顶。当新的 Activity 启动入栈时，原 Activity 会被压入到栈的第二层。一个 Activity 在栈中的位置变化反映了它在不同状



态间的转换。

4. Activity 生命周期

在 android.app.Activity 类中，Android 定义了一系列与生命周期相关的方法，在我们自己的 Activity 中，只是根据需要复写相应的方法，Java 的多态性会保证我们自己的方法被虚拟机调用，这一点与 J2ME 中的 MIDlet 类似。

这些方法的说明如下：

protected void onCreate(Bundle savedInstanceState)是一个 Activity 的实例被启动时调用的第一个方法。一般情况下，我们都覆盖该方法作为应用程序的一个入口点，在这里做一些初始化数据、设置用户界面等工作。大多数情况下，我们都要在这里从 XML 中加载设计好的用户界面。

当然，也可从 savedInstanceState 中读取保存到存储设备中的数据，但是需要判断 savedInstanceState 是否为 null，因为 Activity 第一次启动时并没有数据被存储在设备中。

protected void onStart 方法会在 onCreate 方法之后被调用，或者在 Activity 从 Stop 状态转换为 Active 状态时被调用。

protected void onResume 方法会在 Activity 从 Pause 状态转换到 Active 状态时被调用。

protected void onResume 方法会在 Activity 从 Active 状态转换到 Pause 状态时被调用。

protected void onStop 方法会在 Activity 从 Active 状态转换到 Stop 状态时被调用。一般在这里保存 Activity 的状态信息。

protected void onDestroy 方法会在 Active 被结束时调用，也调用的最后一个方法，在这里一般做些释放资源、清理内存等工作。

此外，Android 还定义了一些不常用的、与生命周期相关的方法，Android 提供的文档详细地说明了它们的调用规则。

综合实训七　服务器的设计与实现

【问题情境】

在介绍了微博随身 Android 端的功能结构之后，从现在开始将对其具体实现进行介绍，Android 手机端必须要有服务器的支持，本实训将介绍 Android 端服务器的设计与实现。

【拓展知识】

1. 服务器的设计

本系统中，由于 Android 手机端的服务器和 Web 服务器都需要调用 DBUtil 类的业务方法，因此手机端的服务器和 Web 服务器是绑定在一起的。将启动手机端服务器的代码写到 Web 容器的 SevletContextListener 监听器中，当 Web 容器启动时就会指定启动手机端服务器的代码。



手机端的服务器启动后，会创建一个 ServerSocket 监听指定的端口，同时还会启动一个线程 ServerThread 负责监听用户的连接，每当有客户端进行连接时，会创建并启动一个 ServerAgent 进程对象专门负责与该客户端进行通信。

2. 服务器的实现

前面已经对手机端服务器的设计进行了介绍，下面将说明其具体的开发过程，步骤如下。

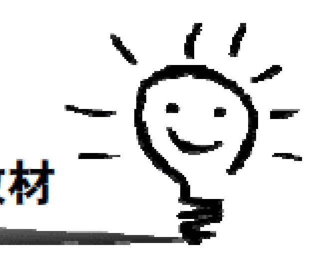
（1）在微博随身 Web 端服务器项目中新建一个继承自 ServletContextListener 的 Java 类 MyServletContextListener，其目录位置与 Web 端 DBUtil 等其他 Java 类文件相同，其代码如下。

```
1    package wpf;                                         //声明包语句
2    import java.net.ServerSocket;                        //引入相关类
3    import javax.servlet.ServletContextEvent;            //引入相关类
4    import javax.servlet.ServletContextListener;         //引入相关类
5    public class MyServletContextListener implements ServletContextListener{
6            ServerSocket ss = null;                      //声明 ServerSocket 对象
7            ServerThread st = null;                      //声明 ServerThread 对象
8           public void contextInitialized(ServletContextEvent sce){   //重写 contextInitialized 方法
9                   try{
10                          ss = new ServerSocket(8888);  //创建 ServerSocket 对象
11                          st = new ServerThread(ss);    //创建 ServerThread
12                          st.start();                   //启动 ServerThread
13                  }catch(Exception e){
14                          e.printStackTrace();          //捕获并打印异常
15                  }
16           }
17           public void contextDestroyed(ServletContextEvent sce){ //重写 contextDestroyed 方法
18                  try{
19                          st.flag = false;              //设置 ServerThread 的标志位
20                          ss.close();                   //关闭 ServerSocket
21                          ss = null;                    //将 ss 置空
22                          st = null;                    //将 st 置空
23                  }catch(Exception e){
24                          e.printStackTrace();          //捕获并打印异常
25                  }
26           }
27   }
```

☑ 第 6 行和第 7 行分别声明了 ServerSocket 对象和 ServerThread 对象。

☑ 第 8～16 行为重写的 contextInitialized 方法，该方法将在 Web 容器（Tomcat）启动时被调用。重写的 contextInitializet 方法主要进行的工作是创建一个 ServerSocket 对象，据此创建一个 ServerThread 线程并启动该线程。

☑ 第 17～26 行为重写的 contextDestroyed 方法，该方法将在 Web 容器关闭时被调用。该方法进行的主要工作是停止 ServerThread 的执行并关闭 ServerSocket。



（2）新建一个 ServerThread.java 文件，其文件所在位置与 MyServletContextListener 相同，在其中输入如下代码。

```
1   package wpf;                                  //声明包语句
2   import java.net.ServerSocket;                 //引入相关类
3   import java.net.Socket;                       //引入相关类
4   public class ServerThread extends Thread{
5           public ServerSocket ss;               //声明 ServerSocket 对象
6           public boolean flag = false;          //声明控制线程执行的标志位
7           public ServerThread(ServerSocket ss){ //构造器
8                   this.ss = ss;
9                   flag = true;                  //设置线程执行标志位
10          }
11          public void run(){                    //线程执行方法
12              while(flag){                      //循环
13                  try{
14                          Socket socket = ss.accept();                     //等待连接
15                          ServerAgent sa = new ServerAgent(socket);        //创建 ServerAgent
16                          sa.start();                                      //启动 ServerAgent 对象
17                  }catch(Exception e){
18                          e.printStackTrace();                             //捕获并打印异常
19  }}}}
```

☑ 第 5 行声明了一个 ServerSocket 对象的引用，该对象将会在代码第 7～10 行的构造器中被初始化。

☑ 第 11～19 行为线程的 run 方法的代码，主要进行的操作为获得客户端的 Socket 连接，将其作为 ServerAgent 的构造器参数创建 ServerAgent 线程并启动 ServerAgent。

（3）新建一个 ServerAgent.java 文件，其文件位置与 MyServletContextListener 相同，在其中输入如下代码。

```
package wpf;                              //声明包语句
import static wpf.ConstantUtil.*;         //引入相关类
import java.io.DataInputStream;           //引入相关类
...                                       //此处省略部分引入相关类的代码
import java.util.ArrayList;               //引入相关类
import java.util.List;                    //引入相关类
public class ServerAgent extends Thread{
        public Socket socket;             //声明 Socket 对象
        public DataInputStream din;       //声明输入流对象
        public DataOutputStream dout;     //声明输出流对象
        boolean flag = false;             //声明线程执行标志位
```



```
12         public ServerAgent(Socket socket){      //构造器
13             this.socket = socket;
14             try {
15                 this.din = new DataInputStream(socket.getInputStream());      //获得输入流对象
16                 this.dout = new DataOutputStream(socket.getOutputStream());  //获得输出流对象
17                 flag =true;                                                 //设置线程执行标志位
18             }
19             catch (IOException e) {e.printStackTrace();}
20             public void run(){                                              //方法：线程执行方法
21                 while(flag){
22                     try {
23                         String msg = din.readUTF();                         //接收客户端发来的消息
24                         if(msg.startsWith("<#LOGIN#>")){                   //消息为登录
25                         ...//此处声明消息类型为登录时的处理代码
26                         }
27                     ...//此处省略消息类型为其他类别时的处理代码
28                     }
28                     catch(EOFException eof){                                //捕获 EOFException
29                      try {
30                          dout.close();din.close();socket.close();           //关闭输入、输出流及 Socket 对象
31                          socket = null;flag = false;                        //停止线程执行
32                      } catch (IOException e) {e.printStackTrace();}//捕获并打印异常
33                 }
34                 catch (Exception e) {e.printStackTrace();}                 //捕获并打印异常
35                }
36         }
37     }
```

☑ 第 8～11 行为 ServerAgent 类成员变量的声明。这些成员变量将在代码第 12～19 行的构造器中被初始化。

☑ 第 20～36 行为 ServerAgent 线程的 run 方法的代码。在 run 方法的每一次循环中，都将接收来自客户端的消息，并根据消息源的不同而执行不同的处理代码。第 24～26 行以登录消息为例说明了这个问题。

☑ 第 29～33 行为捕获 EOFException 异常并进行相应处理的代码，当 ServerAgent 运行时抛出该异常时，将执行第 30～31 行的代码以停止执行 ServerAgent。

提示

代码第 27 行省略了 ServerAgent 对其他类型的客户端消息进行处理的代码，这些代码将会在后面介绍特定功能时进行详细介绍，在介绍时将只列出具体的处理代码，而不会重复列出 ServerAgent 类的框架代码。



📖 小知识十：什么是 Web 服务器

通俗地讲，Web 服务器是传送（serves）页面使浏览器可以浏览，然而应用程序服务器提供的是客户端应用程序可以调用（call）的方法（methods）。确切地说，Web 服务器专门处理 HTTP 请求（request），但是应用程序服务器是通过很多协议来为应用程序提供（serves）商业逻辑（business logic）。

（1）Web 服务器（Web Server）

Web 服务器可以解析（handles）HTTP 协议。当 Web 服务器接收到一个 HTTP 请求时（request），会返回一个 HTTP 响应（response），例如，送回一个 HTML 页面。为了处理一个请求（request），Web 服务器可以响应一个静态页面或图片，进行页面跳转（redirect），或者把动态响应（dynamic response）的产生委托（delegate）给一些其他的程序，例如，CGI 脚本、JSP（JavaServer Pages）脚本、servlets、ASP（Active Server Pages）脚本、服务器端（server-side）JavaScript 或者一些其他的服务器端（server-side）技术。无论它们（译者注：脚本）的目的如何，这些服务器端（server-side）的程序通常产生一个 HTML 的响应来让浏览器可以浏览。

要知道，Web 服务器的代理模型（delegation model）非常简单。当一个请求被送到 Web 服务器时，它只单纯地把请求传递给能很好地处理请求的程序（译者注：服务器端脚本）。Web 服务器仅仅提供一个可以执行服务器端（server-side）程序和返回（程序所产生的）响应的环境，而不会超出职能范围。服务器端程序通常具有事务处理（transaction processing）、数据库连接（database connectivity）和消息（messaging）等功能。

虽然 Web 服务器不支持事务处理或数据库连接池，但它可以配置（employ）各种策略（strategies）来实现容错性（fault tolerance）和可扩展性（scalability），例如，负载平衡（load balancing）、缓冲（caching）。集群特征（clustering—features）经常被误认为仅仅是应用程序服务器专有的特征。

（2）应用程序服务器（The Application Server）

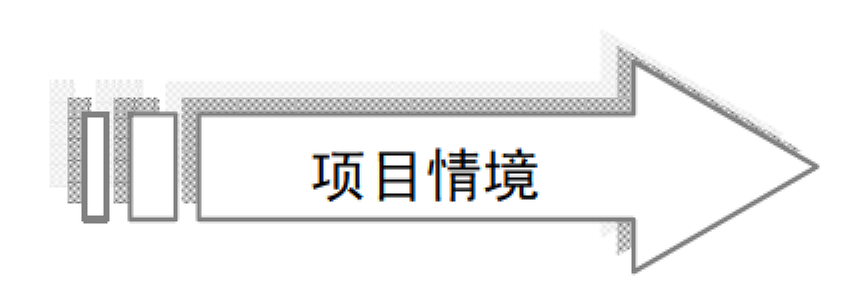
根据定义，作为应用程序服务器，它通过各种协议（包括 HTTP），把商业逻辑暴露给（expose）客户端应用程序。Web 服务器主要是处理向浏览器发送 HTML 以供浏览，而应用程序服务器提供访问商业逻辑的途径以供客户端应用程序使用。应用程序使用此商业逻辑就像你调用对象的一个方法（或过程语言中的一个函数）一样。

应用程序服务器的客户端——包含有图形用户界面（GUI）的——可能会运行在一台 PC、一个 Web 服务器甚至是其他应用程序的服务器上。在应用程序服务器与其客户端之间来回穿梭（traveling）的信息不仅仅局限于简单的显示标记。相反，这种信息就是程序逻辑（program logic）。正是由于这种逻辑取得了（takes）数据和方法调用（calls）的形式而不是静态 HTML，所以客户端才可以随心所欲地使用这种被暴露的商业逻辑。

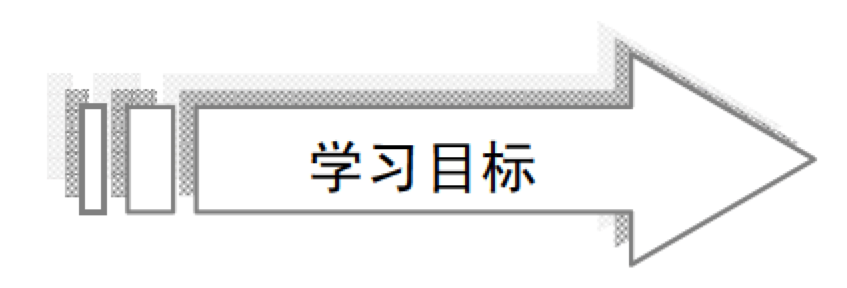
在大多数情形下，应用程序服务器是通过组件（component）的应用程序接口（API）把商业逻辑暴露（expose）给客户端应用程序，例如，基于 J2EE（Java 2 Platform, Enterprise Edition）应用程序服务器的 EJB（Enterprise JavaBean）组件模型。此外，应用程序服务器

也可以管理自己的资源，例如，看大门的工作（gate-keeping duties）包括安全（security）、事务处理（transaction processing）、资源池（resource pooling）和消息（messaging）。就像 Web 服务器一样，应用程序服务器配置了多种可扩展（scalability）和容错（fault tolerance）技术。

项目八
数据存取模块的开发



本项目将对游戏的数据存取模块进行介绍，该模块主要负责对游戏数据的存储与读取，其主要包括地图文件的加载以及游戏存档的恢复。



- 掌握城池信息以及地图层信息封装类的开发，包括城池信息类 CityInfo、地图底层 Layer、地图上层 MeetableLayer 和地图层管理类 LayerList
- 掌握数据存取相关类的开发，包括 GameData 类和 SerializableGame 类

任务 16　城池信息以及地图层信息的封装类

【任务情境】

本任务主要对城池信息的封装类 Cityinfo、地图层信息的封装类 Layer 以及层的其他相关类进行介绍，使读者了解本游戏的数据管理方式。

【相关知识】

1. 城池信息类 CityInfo

CityInfo 类主要封装了城池的静态信息，包括城池名称、所属国家、兵力、粮草、武将列表等，其代码如下。

```
package wyf.ytl;
import java.util.ArrayList;                //引入相关类
import static wyf.ytl.ConstantUtil.*;      //引入 ConstantUtil 中的静态成员
/*该类的成员变量，分别存储着城池的相关信息，例如，城池名称、武将列表、箭垛个数等*/
public class CityInfo {
        String cityName;                   //城池名称
        int country;                       //所属国家
        int army;                          //兵力
        int food;                          //粮草
        int level;                         //等级
        int baseAttack = 20;               //每个士兵的基础攻击力
        int baseDefend = 15;               //每个士兵的基础防御力
        int citizen;                       //居民
        int warTank = 0;                   //战车的个数，用于增加攻城力
        int warTower = 0;                  //箭垛的个数用于增加防御力
        final int info;
        ArrayList<General> guardGeneral = new ArrayList<General>();      //武将列表
/*CityInfo 的构造器，在构造器中将城池的信息进行初始化*/
public CityInfo(String cityName, int country, int army, int food, int level, int baseAttack, int baseDefend,
int citizen, int warTank, int warTower, General guardGeneral,int info) {          //构造器
                this.cityName = cityName;                                          //得到城池名
                this.country = country;
                this.army = army;
                this.food = food;
                this.level = level;
                this.baseAttack = baseAttack;
                this.baseDefend = baseDefend;
                this.citizen = citizen;
                this.warTank = warTank;                                            //得到战车个数
```

```
            this.warTower = warTower;
            this.guardGeneral.add(guardGeneral);
            this.info = info;
        }
    /*以下为恢复城池信息到默认值的方法，通过对常量类中的 CITY_INFO 进行循环，得到当前城池的
信息，然后进行恢复*/
        public void setBackToInit(){                          //恢复到默认
            for(int i=0; i<CITY_INFO.size(); i++){            //循环 ConstantUtil 中 CITY_INFO
                CityInfo ci = CITY_INFO.get(i);               //得到每个 CityInfo
                if(ci.info == this.info){                     //判断是否为当前这个
                    this.cityName = ci.cityName;              //恢复城池名
                    this.country = ci.country;
                    this.army = ci.army;
                    this.food = ci.food;
                    this.level = ci.level;
                    this.baseAttack = ci.baseAttack;
                    this.baseDefend = ci.baseDefend;
                    this.citizen = ci.citizen;
                    this.warTank = ci.warTank;
                    this.warTower = ci.warTower;
                    this.guardGeneral = ci.guardGeneral;      //恢复武将
                    break;                                    //中断循环
                }
            }
        }
    }
```

2. 地图底层 Layer 的介绍

地图底层 Layer 为底层地图封装类，除了存储表示地图信息的 MyDrawable 数组外，还提供了可遇矩阵的计算方法，其代码如下。

```
    package wyf.ytl;
    import java.io.Serializable;                       //引入相关类
    import android.content.res.Resources;              //引入相关类
    public class Layer implements Serializable{        //实现了 Serializable 接口
        private static final long serialVersionUID = -9052906322948980410L;      //持久化版本序列号
        private MyDrawable[][] mapMatrix;              //表示地图的二维数组，该二维数组中存放的是
MyDrawable 对象
        /*该类的两个构造器，因为需要将该类的对象进行序列化，所以必须有以下的空构造器，以及有
参构造器中对该层地图信息进行初始化*/
        public Layer(){}                               //空构造器
        public MyDrawable[][] getMapMatrix(){          //mapMatrix 的 get 方法
            return mapMatrix;
        }
```

```
        public Layer(Resources resources){              //构造器
            this.mapMatrix = GameData.mapData;
        }
    /*计算该层地图的可通过矩阵，通过对地图数组进行循环，得到每个 MyDrawable 对象，然后得到
MyDrawable 所有的不可通过点，并记录到 int 型数组中*/
        public int[][] getNotIn(){                          //得到可通过矩阵
            int[][] result = new int[40][60];               //创建一个 int 型二维数组
            for(int i=0; i<mapMatrix.length; i++){
                for(int j=0; j<mapMatrix[i].length; j++){
                    int x = mapMatrix[i][j].col - mapMatrix[i][j].refCol;
                    int y = mapMatrix[i][j].row + mapMatrix[i][j].refRow;
                                                            //通过占位点及在大地图中的点计算
                    int[][] notIn = mapMatrix[i][j].noThrough;
                    for(int k=0; k<notIn.length; k++){              //对不可通过数组循环
                        result[y-notIn[k][1]][x+notIn[k][0]] = 1;   //不可通过点置 1
                    }
                }
            }
            return result;
        }
    }
```

3. 地图上层 MeetableLayer 的介绍

该类继承自 Layer，为地图上层的封装类，除了包含与底层同样的信息外，还包含了地图的可遇矩阵，其代码如下。

```
    package wyf.ytl;
    import java.io.Serializable;                                        //引入相关类
    import android.content.res.Resources;                               //引入相关类
    public class MeetableLayer extends Layer implements Serializable{   //实现 Serializable 接口
        private static final long serialVersionUID = 1265133035026861860L;  //持久化版本序列号
        private MyMeetableDrawable[][] mapMatrixMeetable;               //实际地图
        private MyMeetableDrawable[][] mapMatrixForMeetable;            //可遇矩阵
       /*上层地图的可遇矩阵，该二维数组中存储的并不是 0 和 1，而是可遇的实体，这样的好处是当
检测可遇成功后，即可直接将遇到的实体返回*/
        public MeetableLayer(){}                                        //空构造器
       /*该类的构造方法，在构造方法中得到上层地图信息并计算其可遇矩阵*/
        public MeetableLayer(Resources resources) {                     //构造器
            super(resources);
            this.mapMatrixMeetable = GameData2.mapData;                 //得到地图信息
            initMapMatrixForMeetable();                                 //计算可遇矩阵
        }
```

```
    */
    public MyDrawable[][] getMapMatrix()                    //计算可遇矩阵
    {
        return mapMatrixMeetable;
    }
```

/*计算可遇矩阵的方法，同样是对地图数组进行循环，得到每个可遇的实体 MyMeetableDrawable，然后根据其占位点以及在地图中的位置计算可遇矩阵*/

```
public void initMapMatrixForMeetable(){//计算可遇矩阵 mapMatrixForMeetable
        mapMatrixForMeetable = new MyMeetableDrawable[40][60];
        for(int i=0; i<mapMatrixMeetable.length; i++){
            for(int j=0; j<mapMatrixMeetable[i].length; j++){
                if(mapMatrixMeetable[i][j] != null){
                    int x = mapMatrixMeetable[i][j].col - mapMatrixMeetable[i][j].refCol;
                    int y = mapMatrixMeetable[i][j].row + mapMatrixMeetable[i][j].refRow;
                    int[][] meetableMatrix = mapMatrixMeetable[i][j].meetableMatrix;
                    for(int k=0; k<meetableMatrix.length; k++){
                    mapMatrixForMeetable[y-meetableMatrix[k][1]][x+meetableMatrix[k][0]]=
                    mapMatrixMeetable[i][j];
                    }
                }
            }
        }
    }
```

/*检测是否遇上，判断英雄的上、下、左、右 4 个方向是否有可遇的实体，如果有，则将该实体返回*/

```
    public MyMeetableDrawable check(Hero hero){             //检测是否遇上
        int col = hero.col;                                 //获取英雄的列数
        int row = hero.row;                                 //获取英雄的行数
        switch(hero.direction%4){                           //还是先按方向查看
        case 0:                                             //向下
            if(mapMatrixForMeetable[row][col-1] != null){   //左边检测到了可遇物
                return mapMatrixForMeetable[row][col-1];
            }
            else if(mapMatrixForMeetable[row][col+1] != null){//右边检测到了可遇物
                return mapMatrixForMeetable[row][col+1];
            }
            break;
        …//该处省略了其他 3 个方向的计算
       }
       return null;                                         //如果没有检测到则返回 null 值
    }
}
```

4. 地图层管理类 LayerList 的介绍

为了管理方便，本游戏开发了 LayerList 类来管理各个地图层，LayerList 包含了各个层的引用以及计算总不可通过矩阵的方法，其代码如下。

```
package wyf.ytl;
import java.io.Serializable;                          //引入相关类
import java.util.ArrayList;                           //引入相关类
import android.content.res.Resources;                 //引入相关类
public class LayerList implements Serializable{       //实现 Resources 接口
    private static final long serialVersionUID = 3249868473800473202L;//持久化版本序列号
    ArrayList<Layer> layers = new ArrayList<Layer>();
                                        //存储 Layer 的容器，存储的是各个层的引用
    public LayerList(){}                              //空构造器
/*该类的构造方法，在方法中调用初始化方法初始化相关资源*/
public LayerList(Resources resources){                //构造器
        this.init(resources);
    }
/*以下的初始化方法中，初始化底层以及上层地图，并将其引用存储到 layers 容器中*/
public void init(Resources resources){                //初始化资源
        Layer l = new Layer(resources);
        layers.add(l);                                //添加底层
        MeetableLayer ml = new MeetableLayer(resources);
        layers.add(ml);                               //添加上层
    }
/*计算总不可通过矩阵的方法，需要对每层地图进行循环，得到每层地图的不可通过矩阵，然后对各
个层的不可通过矩阵进行或运算，得到总不可通过矩阵*/
    public int[][] getTotalNotIn(){                   //得到总不可通过矩阵
        int[][] result = new int[40][60];
        for(Layer layer : layers){                    //对所有层进行循环
            int[][] tempNotIn = layer.getNotIn();
            for(int i=0; i<tempNotIn.length; i++){    //然后对不可通过矩阵循环
                for(int j=0; j<tempNotIn[i].length; j++){
                    result[i][j] = result[i][j] | tempNotIn[i][j];//或运算，得到总不可通过点
                }
            }
        }
        return result;
    }
}
*/
```

任务 17　数据存取相关类的介绍

【任务情境】

本任务将对与数据存储有关的 3 个类进行介绍，其中 GameData 类负责读取底层地图的数据，GameData2 类负责读取上层地图的数据，SerializableGame 类负责游戏存档的存储与读取。

【相关知识】

1．GameData 类的介绍

GameData 类主要负责读取之前底层地图设计器所生成的地图文件中的地图信息，并将其分析成游戏中可用的信息，其代码如下。

```
package wyf.ytl;
import java.io.DataInputStream;              //引入相关类
import java.io.IOException;                  //引入相关类
import java.io.InputStream;                  //引入相关类
import android.content.res.Resources;        //引入相关类
import android.graphics.Bitmap;              //引入相关类
import android.graphics.BitmapFactory;       //引入相关类
/*图片资源的引用，因为图片只需加载一次，所以将其声明成静态成员*/
public class GameData{                       //各种 MyDrawable 对象在这里初始化
    static Resources resources;              //resources 的引用
    static Bitmap grassBitmap;               //草地图片
    static Bitmap xiaoHua1Bitmap;            //小花 1 图片
    static Bitmap muZhuangBitmap;            //木桩图片
    static Bitmap xiaoHua2Bitmap;            //小花 2 图片
    static Bitmap roadBitmap;                //道路图片
    static Bitmap jingBitmap;                //井的图片
    static Bitmap[] bitmaps;                 //图片数组
    public static MyDrawable [][] mapData;   //地图矩阵
//底层地图的二维矩阵，其他类需要绘制或使用底层地图时，必须从此处获得
/*初始化图片资源的方法，在方法中先将所有图片进行初始化，然后再将图片的引用存放到数组中便于管理*/
public static void initMapImage(){           //初始化图片资源的方法
        grassBitmap = BitmapFactory.decodeResource(resources, R.drawable.caodi);
        xiaoHua1Bitmap = BitmapFactory.decodeResource(resources, R.drawable.hua1);
        muZhuangBitmap = BitmapFactory.decodeResource(resources, R.drawable.muzhuang);
        xiaoHua2Bitmap = BitmapFactory.decodeResource(resources, R.drawable.hua2);
        roadBitmap = BitmapFactory.decodeResource(resources, R.drawable.gonglu);
```

```
            jingBitmap = BitmapFactory.decodeResource(resources, R.drawable.jing);
            bitmaps=new Bitmap[]{              //图片数组
                grassBitmap,                   //草地图片的引用
                xiaoHua1Bitmap,                //小花 1 图片的引用
                muZhuangBitmap,                //木桩图片的引用
                xiaoHua2Bitmap,                //小花 2 图片的引用
                roadBitmap,                    //道路图片的引用
                jingBitmap                     //井图片的引用
            };
        }
```

/*读取地图信息的方法，在方法中先创建 MyDrawable 的二维数组用来表示底层地图，然后通过 getAsserts 方法打入输入流，并从文件夹中读取地图信息。此处读取的顺序必须与地图设计器设计地图时的保存顺序完全相同。读取完信息后，根据这些信息创建 MyDrawable 对象并存储到地图数组中

```
    public static void initMapData(){//初始化地图数组
    //二维数组为一个 MyDrawable 对象的引用
        mapData = new MyDrawable [40][60];
        try {
            InputStream in = resources.getAssets().open("maps.so");  //得到输入流
            DataInputStream din = new DataInputStream(in);           //得到数据流
            int totalBlocks = din.readInt();                         //读取实体总共的个数
            for(int i=0; i<totalBlocks; i++){
                int outBitmapInxex = din.readByte();
                int kyf=din.readByte();                              //可遇否，0-不可遇
                int w = din.readByte();                              //图元的宽度
                int h = din.readByte();                              //图元的高度
                int col = din.readByte();                            //总列数
                int row = din.readByte();                            //总行数
                int pCol = din.readByte();                           //占位列
                int pRow = din.readByte();                           //占位行
                int bktgCount=din.readByte();                        //不可通过点的数量
                int[][] notIn=new int[bktgCount][2];
                for(int j=0; j<bktgCount; j++){                      //读入不可通过点
                    notIn[j][0] = din.readByte();
                    notIn[j][1] = din.readByte();
                }
                mapData[row][col]=new MyDrawable(
                        bitmaps[outBitmapInxex],                     //图片
                        ((kyf==0)?false:true),                       //可遇否标志位
                        w, h, col, row, pCol, pRow, notIn            //其他信息
                    );
            }
                din.close();                                         //关闭输入流
                in.close();
                } catch (IOException e) {                            //捕获异常
                e.printStackTrace();                                 //打印异常信息
                }
```

```
        }
}
```

GameData2 类的实现与 GameData 基本相同，只是其读取的是 mapsu.so 文件中的信息，创建的是 MyMeetableDrawable 类的子类对象，因篇幅有限，在此就不再赘述。

2. SerializableGame 类的介绍

SerializableGame 类是用于保存与读取游戏存档的类，该类中包含存档、读取以及检测存档文件是否存在 3 个方法，其开发步骤如下。

（1）开发其代码框架如下。

```
package wyf.ytl;
import java.io.InputStream;                              //引入相关类
import java.io.ObjectInputStream;                        //引入相关类
import java.io.ObjectOutputStream;                       //引入相关类
import java.io.OutputStream;                             //引入相关类
import java.util.ArrayList;                              //引入相关类
import android.media.MediaPlayer;                        //引入相关类
public class SerializableGame {
        public SerializableGame(){}
        public static void saveGameStatus(GameView gameView){
        …//该处省略了游戏保存的代码，将在后面给出
        }
        public static void loadingGameStatus(GameView gameView){
        …//该处省略了游戏加载的代码，将在后面给出
        public static Boolean check(HDZGActivity h){     //检查文件是否存在
            try  {
                h.openFileInput("game.ytl");             //打开文件流
            }catch(Exception e){                         //当捕获到异常时
                return false;                            //返回 false
            }
            return true;                                 //能正常打开时返回 true
        }
}
```

说明

其中 saveGameStatus 用于将当前游戏状态存档，而 loadingGameStatus 方法则是恢复游戏状态到存盘点。

（2）对 saveGameStatus 方法进行完善，其代码如下。

```
public static void saveGameStatus(GameView gameView){    //保存游戏的方法
    OutputStream out = null;                             //输出流
    ObjectOutputStream oout = null;                      //声明 ObjectOutputStream 的引用
```



```
        try{
            out = gameView.getContext().openFileOutput("game.ytl", 0);   //打开文件流
            oout = new ObjectOutputStream(out);
            oout.writeObject(gameView.hero);                  //保存英雄对象
            oout.writeInt(gameView.startRow);                 //屏幕在大地图中的行数
            oout.writeInt(gameView.startCol);                 //屏幕在大地图中的列数
            oout.writeObject(gameView.skillToLearn);          //将要学习的技能
            oout.writeObject(gameView.allCityDrawable);       //存放所有敌方的城池
            oout.writeObject(gameView.freeGeneral);           //自由的将领，即不属于我方和敌方的
            oout.close();
            out.close();                                      //关闭相关流
        }catch(Exception e){                                  //捕获异常
            e.printStackTrace();                              //打印异常信息
        }
    }
```

说明

该方法只是通过 ObjectOutputStream 将需要存储的游戏数据写入到文件 game.ytl 中，但前提是被序列化的各个类必须实现 Externalizable 或 Serializable 接口，而实现 Externalizable 接口的类还需实现接口中的两个抽象方法来完成数据的存储和读取。

（3）接下来对 loadingGameStatus 方法进行介绍，其代码如下。

```
public static void loadingGameStatus(GameView gameView){      //加载游戏的方法
    InputStream in = null;                                    //声明 InputStream 的引用
    ObjectInputStream oin = null;                             //声明 ObjectInputStream 的引用
    try{
        /*初始化相关输入流*/
        in = gameView.getContext().openFileInput("game.ytl"); //得到输入流
        oin = new ObjectInputStream(in);                      //初始化数据流
        /*停止所有相关线程*/
        gameView.hero.ht.flag=false;                          //停止 ht 线程
        gameView.hero.ht.isGameOn=false;
        gameView.hero.ht.interrupt();
        gameView.hero.hbdt.flag=false;                        //停止 hbdt 线程
        gameView.hero.hbdt.interrupt();
        if(gameView.activity.loadingView.process<90){
            gameView.activity.loadingView.process = 90;       //走加载界面的进度条
        }
        /*从存档文件中读取之前存储的游戏信息，并将其恢复到游戏中*/
        gameView.hero = (Hero) oin.readObject();
        MyDrawable[][] mba=gameView.layerList.layers.get(1).getMapMatrix();
        /*恢复所有敌方城池的信息*/
        for(CityDrawable c : gameView.allCityDrawable){
```

```
            int colT=c.col;
            …//该处省略了回复城池相关信息的代码
        c.bmpDialogButton=((CityDrawable)mba[rowT][colT]).bmpDialogButton;
            mba[rowT][colT]=c;
          }
        for(CityDrawable c : gameView.hero.cityList){                    //循环
            int colT=c.col;
            …//该处省略了回复城池相关信息的代码
          /*恢复英雄所统治的城池的信息*/
         c.bmpDialogButton=((CityDrawable)mba[rowT][colT]).bmpDialogButton;
         mba[rowT][colT]=c;
         }
    ((MeetableLayer)gameView.layerList.layers.get(1)).initMapMatrixForMeetable();
     gameView.hero.father = gameView;                                 //恢复英雄相关信息
     gameView.hero.initAnimationSegment(GameView.heroAnimationSegments);
     //为英雄动画段列表
     gameView.hero.startAnimation();                                  //启动英雄动画
     gameView.hero.hgt = new HeroGoThread(gameView,gameView.hero)
      /*恢复游戏相关的线程，并启动应该启动的相关线程。当所有信息恢复完成后，向 Activity
发送 Handler 消息通知 Activity 切换用户界面*/
     gameView.drawThread.setIsViewOn(True);                           //设置标志位
     gameView.hero.hbdt=new HeroBackDataThread(gameView.hero);
     gameView.hero.hbdt.start();                                      //启动线程
     if(!gameView.mMediaPlayer.isPlaying()){                          //当音乐没有播放时
          gameView.mMediaPlayer = MediaPlayer.create(gameView.activity,R.raw.backsound);
          gameView.mMediaPlayer.setLooping(true);                     //设置循环播放
          if(gameView.activity.isBackSound){
                  gameView.mMediaPlayer.start();                      //播放声音
          } }
          gameView.activity.loadingView.process = 101;                //进度条
          gameView.activity.myHandler.sendEmptyMessage(100);          //发送 Handler 消息
          oin.close();
          in.close();                                                 //关闭流
     }
     catch(Exception e){                                              //捕获异常
         e.printStackTrace();                                         //打印异常信息
      }
}
```

【项目小结】

本项目介绍了游戏的数据存取模块，该模块作用是对游戏数据的存储和读取。几乎所有的游戏都有对数据存储和读取的过程，只是根据游戏复杂程度的不同，这一过程的复杂程度不同而已。因此，熟练掌握本项目的方法对于编写其他游戏有着重要的意义。



小知识十一：保存游戏数据详解

对于游戏中的数据进行保存，在 Android 中常用以下四种保存方式。

1. SharedPreference

此保存方式适用于简单数据的保存，顾名思义，属于配置性质的保存，不适合用于数据比较大的保存。

优点：简单、方便、适合简单数据的快速保存。

缺点：

① 存储的文件只能在同一包内使用，不能在不同包之间使用。

② 默认将数据存放在系统路径下 /data/data/com.himi/，暂时没有找到放 SD 卡上的方法。

总结：其实本保存方式如同它的名字一样是个配置保存，虽然方便，但只适合存储比较简单的数据。

2. 文件存储（FileInputStream/FileOutputStream）

此保存方式比较适合游戏的保存和使用，可以保存较大的数据，并且不仅能把数据存储在系统中也能将数据保存到 SD 卡中，相对于 SQLite 来说更容易让大家接受。

优点：

① 适合游戏存储，能存储较大数据。

② 不仅能存储到系统中，也能存储到 SD 卡中。

总结：如果对 SQL 不太熟悉的话，那么选择此种方式最为合适。

3. SQLite

此保存方式比较适合游戏的保存和使用，可以保存较大的数据，并且可以将自己的数据存储到文件系统或者数据库当中，也可以将自己的数据存储到 SQLite 数据库当中，也能将数据保存到 SD 卡中。

（1）什么是 SQLite

SQLite 是一款轻量级数据库，它的设计目的是嵌入式，而且它占用的资源非常少，在嵌入式设备中，只需要几百 KB 的存储空间。

（2）SQLite 的特性

- ☑ 轻量级：使用 SQLite 只需要带一个动态库，就可以享受它的全部功能，而且那个动态库的尺寸相当小。
- ☑ 独立性：SQLite 数据库的核心引擎不需要第三方软件，也不需要所谓的“安装”。
- ☑ 隔离性：SQLite 数据库中所有的信息（如表、视图、触发器等）都包含在一个文件夹内，方便用户管理和维护。
- ☑ 跨平台：SQLite 目前支持大部分操作系统，不仅可在电脑中操作，而且在众多的手机系统中也能够运行，如 Android。
- ☑ 多语言接口：SQLite 数据库支持多语言编程接口。
- ☑ 安全性：SQLite 数据库通过数据库级上的独占性和共享锁来实现独立事务处理。这意味着多个进程可以在同一时间从同一数据库读取数据，但只能有一个可以写入程序。



优点：

① 能存储较多的数据。

② 能将数据库文件存储到 SD 卡中。

（3）什么是 SQLiteDatabase

一个 SQLiteDatabase 的实例代表了一个 SQLite 的数据库，通过 SQLiteDatabase 实例的一些方法执行 SQL 语句，可对数据库进行增、删、查、改的操作。需要注意的是，数据库对于一个应用来说是私有的，并且在同一个应用中数据库的名字也是唯一的。

（4）什么是 SQLiteOpenHelper

顾名思义，这个类是一个辅助类。这个类主要生成一个数据库，并对数据库的版本进行管理。当在程序当中调用这个类的 getWritableDatabase 方法，或 getReadableDatabase 方法时没有数据，那么 Android 系统就会自动生成一个数据库。SQLiteOpenHelper 是一个抽象类，通常需要继承且实现其 3 个函数。

（5）什么是 ContentValues 类

ContentValues 类和 Hashmap、Hashtable 类比较类似，它也是负责存储一些名值对，但是它存储的名值对当中的名是一个 String 类型，而值都是基本类型。

（6）什么是 Cursor

Cursor 在 Android 当中是一个非常有用的接口，通过 Cursor 可以对从数据库查询出来的结果集进行随机读写访问。

4. ContentProvider（不推荐用于游戏保存）

此保存方式不推荐用于保存游戏，虽然此方式不仅能存储较大数据，还支持多个程序之间的数据进行交换。但是由于游戏中基本不会去访问外部应用的数据，所以对于此方式不予讲解，有兴趣的读者可以自行学习。

综合实训八　微博随身之 Android 端的准备工作

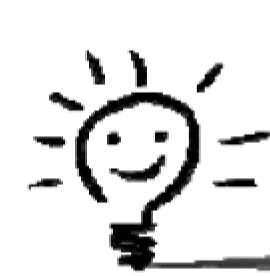
【问题情境】

前面已经介绍了微博随身 Android 端的设计与实现，在正式进入 Android 客户端部分的开发前，需要进行一些准备工作。本实训将从图片资源、XML 资源文件两个方面说明准备工作的内容。

【拓展知识】

1. 图片资源的准备

在 Eclipse 中新建一个 Android 项目 KDWB_Android。在进行开发之前，首先要简单准备一下程序中要用到的图片资源，并将这些图片资源存放在项目文件夹 res\drawable-mdpi 目录下。



2. XML 资源文件的准备

完成了图片资源的准备，下面开始编写程序中的一些 XML 资源文件，主要包括 colors.xml、string.xml 和 style.xml。

（1）colors.xml 的开发

colors.xml 中存放了一些定义好的颜色，这些颜色可以在程序的代码和 XML 布局文件中被调用。在项目的 res\values 目录下新建一个文件 colors.xml，在其中输入以下代码。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="listDivider">#ffcc99</color><!--声明名为 listDivider 的颜色资源-->
    <color name="character">#f37301</color><><!--声明名为 character 的颜色资源-->
</resources>
```

（2）string.xml 的开发

string.xml 文件用于存放字符串资源，项目创建后会默认在 res\values 目录下创建一个 strings.xml，在开发之前和开发过程中会根据需要向其中添加字符串资源，其代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="KDWB">口袋微博</string><!-- 用做标题的字符串资源 -->
    <string name="RegActivity">口袋微博--注册</string>
    <string name="FunctionTab">口袋微博--个人中心</string>
    <string name="PublishActivity">口袋微博--快速发布</string>
    <!-- 此处省略部分 String 资源的声明 -->
    <string name="btnSearch">搜索</string><!-- 用做按钮上显示的文本 -->
    <string name="btnDelete">删除</string>
    <string name="tvInputStatus">更新您的心情：</string><!--用做 TextView 显示内容-->
    <string name="hintStatus">今天心情怎么样？</string>
    <string name="btnEdit">编辑</string>
    <string name="btnPublishMofify">发布修改</string>
</resources>
```



说明

上述代码中声明的字符串资源将主要用于 Activity 标题、按钮显示文本、TextView 及 EditText 空间显示的文本内容。

（3）style.xml 的开发

在项目 res\values 目录下新建一个文件 style.xml，该文件中存放一些定义好的风格样式，这些风格样式是一系列作用于单个控件元素的属性，如定义 TextView 显示的文本内容的颜色、字号大小等。在 style.xml 中定义的风格可以通过在 XML 布局文件中设置控件的“style”属性来实现，也可以在代码中直接设置。本程序中的 style.xml 代码如下。



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="button"><!--定义名为 button 的样式-->
        <item name="android:textSize">20sp</item>
        <item name="android:textColor">#f37301</item>
        <item name="android:textStyle">bold</item>
    </style>
    <style name="title"><!--定义名为 title 的样式-->
        <item name="android:textSize">22sp</item>
        <item name="android:textColor">#f37301</item>
        <item name="android:textStyle">bold</item>
    </style>
    <style name="text"><!--定义名为 text 的样式-->
        <item name="android:textSize">18sp</item>
       <item name="android:textColor">#f37301</item>
        <item name="android:textStyle">bold</item>
    </style>
    <style name="content"><!--定义名为 content 的样式-->
        <item name="android:textSize">16sp</item>
        <item name="android:textColor">#003cda</item>
        <item name="android:textStyle">bold</item>
    </style>
</resources>
```

说明

上述代码中定义了 4 种风格样式，将分别用于设置按钮和不同的 TextView 文本内容的风格样式。

小知识十二：小窥 XML

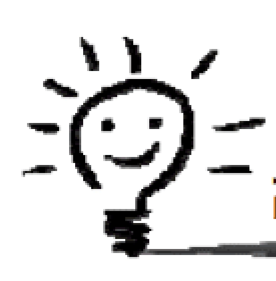
1. XML 文档

什么是格式正规的 XML 文档？所有遵守规定的 XML 基本语法规则的文档（数据）都称为格式正规的 XML 文档。这类数据在使用时可以不用 DTD 或模式来描述它们的结构，它们也称作独立的（或非 DTD）XML 数据，这些数据不能依靠外部的声明，属性值只能是没有经过特殊处理的值或默认值。

一个格式正规的 XML 数据包含一个或多个元素，它们相互之间正确地嵌套，其中有一个元素，即文档元素，包含了文档中其他所有元素，所有的元素构成一棵简单的层次树，所以元素和元素之间唯一的直接关系就是父子关系，兄弟关系经常能够通过 XML 应用程序内部的数据结构推断出来，但这些既不直接也不可靠（因为元素可能被插入到某个元素或多个子元素之间）。其文档内容包括标记和（或）字符数据。

满足下列条件的数据就是格式正规的 XML：

☑ 结束标记匹配相应的起始标记，并且在元素定义中没有重叠，对一个元素来说，没有多个相同名称的属性的实例。



- ☑ XML 语法符合规范，包括起始标记都有匹配的结束标记（空元素标记除外）、元素标记不重叠、属性有唯一的名称、标记字符能被正确地转义。
- ☑ 元素构成一个层次树，只有一个根节点。
- ☑ 没有对外部实体的引用（除非提供了 DTD）。

格式正规的文档可以使用 XML 数据而又不必承担构建和引用数据外部数据描述的重任。术语“格式正规的”与正规的数学逻辑有着相似之处，一个命题如果满足语法规则，那么就是格式正规的，并不在意命题是真是假。

XML 数据对象被认为是有效的 XML 文档的条件是：它必须是格式正规的，并且它能够满足某些进一步有效性约束和匹配描述文档内容的语法。XML 以 XML Schema 或 DTD 的形式提供文档结构的描述。

使用 DTD 进行有效性验证可以确保：服从元素的父子关系，属性具有有效性，所有引用的实体被正确定义，以及遵守其他的有效性约束。

2. XML 的文档类型定义 DTD

DTD 是用来结构化 XML 数据的一套规则，当在一个更广的环境里，如 B2B 或是电子商务系统中，交换、处理及显示 XML 时，定义这样的规则是很重要的，使用 DTD 不仅允许 XML 数据遵循 XML1.0 的一些高级的语法规则，也允许数据在内容和结构方面遵循如下一些我们自定义的规则。

1）使用 DTD 的必要性

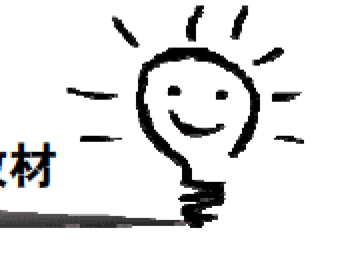
（1）为什么要验证 XML 的有效性？

① 格式正规的 XML 数据要保证对 XML 语法和嵌套（分层）树结构的正确使用，这样可充分地关联静态内部应用程序，特别是在 XML 数据是由计算机产生和处理的时候。在这种情况下，应用程序有责任使用数据完成执行任何结构或者内容的验证、错误的处理和数据的解释。XML 结构化信息及其逻辑，在发送和接收应用程序中通常和一般的规范不一样，通常是通过硬编码独立完成的，因此，更改 XML 数据结构必须从规范、发送和接收应用程序三个方面入手。

② 例如，当一个内部应用程序在两个不同的关系数据库管理系统之间用格式正规的 XML 作为数据传输机制时，发送端假定已准备好了数据，而接收机也具备了确定输入数据的功能，这样任何数据验证都将发生在 XML 到数据库管理系统的传输过程之后。当数据在 XML 域时，不需要重新确定，因此，格式正规的 XML 能胜任数据传输。如果没有正确地描述 XML 数据，那么描述或修改它的数据结构就困难了，因此它的结构和内容都压缩在应用程序的代码里，我们仅能利用 XML 的一小部分功能。

③ 除了保证 XML 数据的格式规范之外，我们并不是每次都要保证 XML 应用程序是有效的 XML 数据。为了做到这点，我们需要：

- ☑ 描述和确定数据结构，最好使之符合严格的、正规的风格。
- ☑ 传送这些数据结构给其他的应用程序和用户。
- ☑ 限制元素内容。



☑　限制属性类型和值，尽可能提供默认值。

④ 这些功能能够在两个正在协作的应用程序和伴随它们的文档中通过特定的代码进行处理。然而，在多个应用程序和用户之间，想在每一个应用程序中维护这些功能将成为一件非常困难的事，因此要寻找一种更标准的方法。

⑤ 从独立的应用程序中分离出 XML 数据描述，允许所有协作应用程序共享一个单独的数据描述，这就是 XML 词汇表。共享通用的 XML 词汇表的一组 XML 文档被称为文档类型，而遵从某一文档类型的一个单独的文档叫文档实例。

2）有效的 XML 是格式正规的数据，这种数据遵循语法、结构和其他的一些在 DTD 中定义的规则。

① 可使用 XML 解析器来确认一个 XML 文档格式是否正确：许多解析器即是有效性验证解析器，能够提供更严格的验证选项去检查 XML 文档内容是否有效，这表示解析器本身就能够验证文档是否遵循特殊的 XML 词汇规则，这种验证是通过比较文档内容和 DTD 形式的相关模板来完成的。

② DTD 能被多个文档和应用程序共享，它的 XML 数据的集中描述和标准的有效性验证方法可使我们从众多的专门应用程序中移出数据描述和验证代码。数据描述代码成了 DTD，有效性验证代码已存在于有效性验证 XML 解析器中，极大地简化了应用程序代码，提高了其性能和可靠性。

2）DTD 的详细描述

（1）定义

① DTD 是一组能融合在 XML 数据里或者在单独的文档存在的声明，它定义一些规则来描述结构和被允许的 XML 数据内容，一个 DTD 只能和一个给定的 XML 文档或数据对象关联起来。

② DTD 使用一个正式的文法来描述 XML 文档的结构和语法，包括大量文档内容的允许值，这些被称为有效性约束的规则，确保了任何 XML 数据遵循与之关联的 DTD。

③ DTD 是 XML1.0 规范的一部分，所以 XML 解析器和其他一些工具已广泛支持这种数据描述和有效验证方法。

④ DTD 不使用格式正规的 XML 描述，但 DTD 声明表面上类似于 XML 的语法，用括号来分隔，并且在 DTD 里使用一种实体引用的特殊形式。DTD 里的注释所使用的语法同样也能用在 XML 的注释里。

（2）特征

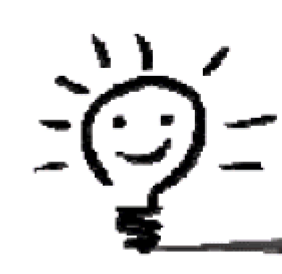
① DTD 有效性验证的最大意义在于对元素分层树的结构定义：一个有效性验证解析器和 DTD 能确保所有必要的元素和属性在文档里出现，并且文档中没有未经授权的元素和属性，这确保了数据在被移交给应用程序之前就具有有效的结构。

② DTD 能用来与一个有效性验证解析器协作，对存在的 XML 数据进行有效性验证或在用户创建 XML 文档的过程中，对文档进行有效性验证，常用方法如下：

☑　对必须出现的元素进行检查。

☑　在使用能读懂 DTD 的 XML 编译器的时候，提示作者包含它们。

☑　确认没有包含被禁止的元素，并且防止作者使用它们。



☑ 实施元素内容和树结构。

☑ 实施元素属性和它们允许的值。

③ 当 DTD 用于 XML 数据实例中的时候，提供信息并简化操作：

☑ 当属性值在 XML 数据中被忽略时，使用默认值。

☑ 可转换内容。

☑ 可简化内容。

④ DTD 还有面向 DTD 作者的附加功能：

☑ DTD 可置换的内容。

☑ 描述非 XML 数据。

（3）结构：一个给定 XML 文档的 DTD 被分成两部分，即内部子集和外部子集，它们是相对于 XML 文档实例来命名的

① 内部子集：是包含于 XML 数据中的 DTD 的一部分。

② 外部子集：是一组定位于各个独立的文档中的声明，它常作为 DTD 引用，因为它经常包含在一个以.dtd 为扩展名的文件中，虽然这是很正常的用法，但外部 DTD 无须是一个单独的文件，它能作为一个记录存储在数据库中，而内部子集也是同一 DTD 中的一部分。

③ 并不要求 DTD 使用子集，DTD 或许能完整地包含于 XML 数据中（内部子集），而没有外部子集，或是一个文档能简单引用外部子集而自身不包含 DTD 声明。在许多情况下，DTD 会将两者结合使用（内部子集和外部子集都使用）。

④ 当相似的声明存在于这两个子集中时，内部子集中的 DTD 优先于外部子集中的 DTD 声明。将会使用来自于内部子集的声明，而来自外部子集的声明将会忽略。

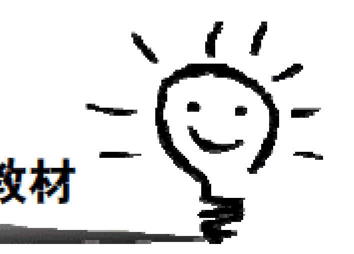
⑤ 例如，当内部子集确定了一个空元素时，一个元素的外部子集描述可以允许有几个子元素。包含此 DTD 联合的任何 XML 数据，最好使用一个空元素，否则将被认为是无效的。

（4）将 DTD 与 XML 数据关联

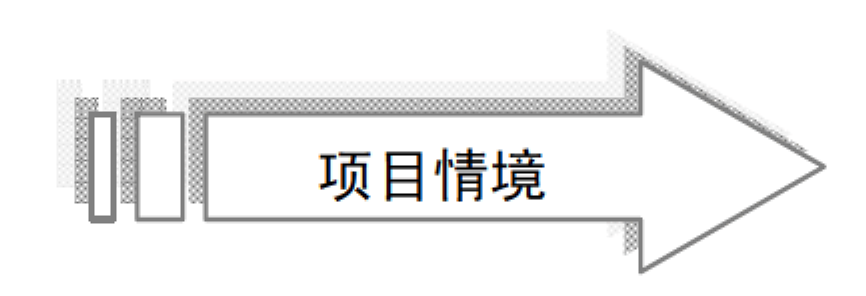
① 用 Document Type Declaration 标记把 DTD 关联到 XML 数据对象，它从不简写成 DTD，通过作为 DOCTYPE 声明引用，以与 DTD 区别。

② 有效性验证解析器正是使用这个声明来检索 DTD，并根据 DTD 规则来验证文档的有效性，如果 DTD 没有找到，那么解析器就会发送一个出错消息，而不能验证文档。

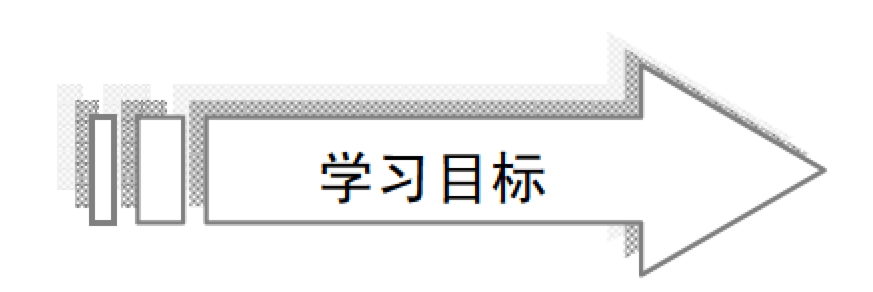
③ DTD 仅仅指一个文档类型的定义，而不是把 DTD 和 XML 文档实例关联在一起的 DOCTYPE 声明。使用单独的 DOCTYPE 声明时，一个 XML 文档只能关联一个 DTD。



项目九
英雄角色模块的开发



在前面已经对该游戏的实体模块进行了简单的介绍，本项目将对实体模块之一——英雄角色模块的开发进行介绍，该模块涉及到的类有 Hero、HeroGoThread、HeroBackDataThread、General 和 Research。



- 学习 Hero 类的代码框架，以及成员变量的声明方法
- 掌握 HeroGoThread 类的开发方法，根据骰子点数移动英雄，并对是否需要转弯、是否需要需要滚屏、是否遇到可遇实体进行检测
- 掌握 HeroBackDataThread 类的开发方法，这是 Hero 类的另一个辅助线程

任务 18　Hero 类的代码框架

【任务情境】

Hero 类是英雄角色实体部分的主要类，本任务先来简单地介绍 Hero 类的代码框架。

【相关知识】

其具体步骤如下。

（1）Hero 类的成员变量的声明，Hero 类中的成员变量大多用于记录游戏的状态，如当前的金钱、兵力等，同时也有与绘制英雄相关的成员变量。Hero 类的成员变量部分的代码如下。

```
package wyf.ytl;                          //声明包语句
import static wyf.ytl.ConstantUtil.*;     //引入相关类
import java.io.Externalizable;            //引入相关类
import java.io.IOException;               //引入相关类
import java.io.ObjectInput;               //引入相关类
import java.io.ObjectOutput;              //引入相关类
import java.util.ArrayList;               //引入相关类
import java.util.HashMap;                 //引入相关类
import java.util.Random;                  //引入相关类
import android.graphics.Bitmap;           //引入相关类
import android.graphics.Canvas;           //引入相关类
/*此类为游戏中一个重要的类，封装了玩家的所有游戏信息，以及一些游戏中需要用到的方法
在该类中既包含英雄的个人属性，也包含其手下的将领列表，拥有的城池列表等*/
public class Hero implements Externalizable{
    public Hero(){}                       //我方可能得到的将领
    public static ArrayList<General> MY_GENERAL = new ArrayList<General>();
    /*自由将领列表声明了存放自由将领的列表，游戏中英雄在招贤馆有可能获得该列表中的
将领*/
    private static final long serialVersionUID = -5433306195939766058L;     //持久化版本序列号
    GameView father;                      //GameView 引用
    private String name = "段世雄";
    int strength = 100;                   //英雄体力，
    int maxStrength = 100;                //体力上限，会随着等级增加
    int level = 1;                        //英雄等级
    int direction = -1;                   //英雄的移动方向，4：下，5：左，6:右，7：上。同时也
代表当前英雄的动画段号，从 0 开始
    int currentFrame = 0;                 //当前英雄的动画段的当前动画帧，从 0 开始
    int food = 10000;                     //随身携带的粮草，不包括每个城池中的粮食
    int totalMoney=6000;                  //总金钱
```

```
        int armyWithMe = 6000;              //跟随自己的军队
        int totalArmy;                      //总军队，包括跟随自己的和守城在家的
```

/*分别声明英雄定位点所在地图中的行和列以及该点在地图中的坐标。本游戏中英雄的像素大小为31×62，英雄的定位点为靠下31×31部分的中心*/

```
        int col;            //英雄的定位点在大地图中的列，定位点为下面的格子的中心
        int row;            //英雄的定位点在大地图中的行，定位点为下面的格子的中心
        int x;              //英雄"中心点"的x坐标，用于绘制
        int y;              //英雄"中心点"的y坐标，用于绘制
        int width;          //英雄的宽度——在initAnimationSegment方法中初始化
        int height;         //英雄的高度——在initAnimationSegment方法中初始化
        int title = 0;      //官衔
```

/*声明了代表英雄官衔的成员变量，当英雄遇到皇城时，有可能通过进贡金钱给天子而获得官衔*/

```
        int warTank = 0;        //随身携带的战车
        int warTower = 0;       //随身携带的箭垛
        int strengthGrowSpan = 1;
```

/*声明的成员变量用于计算游戏体力增加的点数，计算英雄体力增加的方法是用英雄走过的地图格子数乘以该成员变量的值*/

```
        int basicAttack = 18;   //基础攻击力
        int basicDefend = 18;   //基础防御力
```

/*以上两行声明的成员变量将用于计算英雄当前的攻击力和防御力，计算时这两个成员变量将会和当前跟随的兵力做乘积*/

```
        ArrayList<Bitmap []> animationSegment = new ArrayList<Bitmap []>();
                            //存放英雄所有的动画段，每个动画段为一个一维数组
```

/*声明了用于存放英雄拥有城池、将领、科研项目以及技能的集合对象，General 和 Research 类的开发比较简单，只包括一些必要的成员变量和对应的 get 和 set 方法，故本次不再列出其详细代码。Skill 类的开发将在后面的内容中提到*/

```
        ArrayList<CityDrawable> cityList = new ArrayList<CityDrawable>();     //存放英雄拥有的城池
        ArrayList<General> generalList = new ArrayList<General>();            //存放跟随英雄的将领
        ArrayList<Research> researchList = new ArrayList<Research>();         //存放现有的科研项目
        HashMap<Integer,Skill> heroSkill = new HashMap<Integer,Skill>();      //存放英雄的所有技能
        HeroThread ht;              //负责英雄动画换帧的线程
```

/*声明了负责为英雄动画换帧的 HeroThread 对象*/

```
        HeroGoThread hgt;           //负责英雄走路的线程
```

/*声明了 HeroGoThread 对象，该对象负责根据英雄掷出的骰子移动英雄*/

```
        HeroBackDataThread hbdt; //负责后台数据更改的线程，如粮草减少，科研进度/敌人来袭等
```

/*声明了 HeroBackDataThread 对象，该对象负责后台数据更改的线程，如减少粮草、推进科研进度、随机选择城市被偷袭等*/

```
        //构造器
        public Hero(GameView father,int col,int row){
            this.col = col;
            this.row = row;
            this.x = col*TILE_SIZE+TILE_SIZE/2+1;
            this.y = row*TILE_SIZE+TILE_SIZE/2+1;
            this.father = father;
            ht = new HeroThread();
            //
```



（2）Hero 类成员方法的框架，在前面的步骤介绍了 Hero 类的成员变量，下面来介绍 Hero 类的构造器和成员方法的代码框架。

```
public Hero(){}                    //空构造器
/*Hero 类开发了一个无参空实现的构造器，该构造器用于将 Hero 对象进行序列化*/
public Hero(GameView father, int col, int row){//构造器：初始化成员变量}
public void writeExternal(ObjectOutput out) throws IOException{//实现 Externaliza 接口的方法}
public void readExternal(ObjectInput in) throws IOException,ClassNotFoundException{//实现接口方法}
/*以上两行是对 Externalizable 接口的方法 writeExernal 和 readExternal 的实现。关于游戏的存档和读取已在前面进行过介绍，在此将不再赘述*/
/*对英雄的动画帧进行的一系列操作方法*/
public void initAnimationSegment(Bitmap [][] segments){//方法：初始化动画段列表}
public void addAnimationSegment(Bitmap [] segment){//方法：向动画段列表中添加动画段}
public void setAnimationDirection(int direction){//方法：设置方向，同时也将设置动画段}
public void startAnimation(){//方法：开始换帧动画}
public void pauseAnimation(){//方法：暂停动画}
public void destroyAnimation(){//方法：销毁动画及其换帧线程}
public void drawSelf(Canvas canvas,int startRow,int startCol,int offsetX,int offsetY){//方法：绘制自己}
public int [] throwDice(int diceNumber){//掷骰子}
/*throwDice 方法，该方法接收骰子的个数作为参数，然后为每个骰子随机产生一个点数，最后返回一个记录了每个骰子点数的数组*/
public void nextFrame(){//方法：在当前动画段执行换帧操作}
/*对英雄的动画帧进行的一系列操作方法*/
public void startToGo(int steps){//方法：激活英雄的走路线程，传入格子数使英雄开始移动}
/*startToGo 方法，调用该方法将激活 HeroGoThread，然后 HeroGoThread 将根据传入的地图格子数将英雄移动相应的距离*/
public void growStrength(int steps){//方法：根据英雄移动的距离增加英雄的体力}
public void initHeroSkills(){//方法：初始化英雄的技能}
.../*此处省略 Hero 类成员变量的 set 和 get 方法，这些方法的实现方法比较简单，由于篇幅有限，在此将不再赘述*/
```

任务 19　HeroGoThread 类的开发

【任务情境】

HeroGoThread 是 Hero 类的重要辅助线程，其功能主要包括根据骰子点数移动英雄，并对是否需要转弯、是否需要滚屏、是否遇到可遇实体进行检测。

【相关知识】

HeroGoThread 的 run 方法代码框架如下。



```
1   public void run(){                                 //线程执行方法
2           while(flag){
3               while(isMoving){
4                   for(int i=0;i<steps;i++){          //对每一格子进行无级移动
5                    //步骤（1）求出本次移动的目的点
6                    //步骤（2）无级从一个格子走到另一个格子
7               }
8               //步骤（3）检查是否可遇
9           }
10  try{Thread.sleep(waitSpan);                        //线程空转等待
11          catch(Exception e){e.printStackTrace();}//捕获并打印异常
12  }}
```

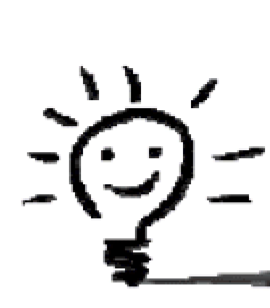
在游戏中，通过设置代码第 3 行用到的 isMoving 变量为 true 来激活 HeroGoThread，代码第 4 行用到的 steps 成员变量代表骰子点数，即要移动的地图格子数。在该线程 run 方法每次循环中，都会按照上述 3 个步骤来进行，下面将对这 3 个步骤进行简单介绍。

（1）首先根据英雄的方向来确定目标点地图格子的行和列，其代码如下。

```
int moves = TILE_SIZE/HERO_MOVING_SPAN;//求出这个格子需要几个小步来完成
/*首先计算出移动一个格子的距离需要几次移动来完成，由于英雄移动的步径并不能整除格子间距离，即图元大小，所以在步骤（2）的最后还需要对英雄的位置进行修正*/
    int hCol = hero.col;                       //英雄当前在大地图中的列
    int hRow = hero.row;                       //英雄当前在大地图中的行
    int destCol=hCol;                          //目标格子列数
    int destRow=hRow;                          //目标格子行数
    /*先求出目的点的格子行和列，之所以模 4 是因为动态向上和静止朝上正好差 4*/
/*通过对英雄当前方向进行判断，计算出目的点的地图格行列数。游戏中英雄可取的方向有 8 种，静止和动态各有 4 个方向，由于动态方向和静态方向之间正好相差 4，所以进行模 4 操作来减少判断次数*/
    switch(hero.direction%4){
    case 0:                                    //向下
        destRow = hRow+1;
        hero.setAnimationDirection(DOWN);      //将英雄的方向和动画段设置为动态向下
        break;
      …//此处省略方向为静（动）太向左和静（动）态向右时的处理代码
    case 3:                                    //向上
        destRow = hRow-1;
        hero.setAnimationDirection(UP);        //将英雄的方向和动画段设置为动态向上
        break;
    }
    int destX=destCol*TILE_SIZE+TILE_SIZE/2+1;  //目的点 x 坐标，已经转换成中心点了
    int destY=destRow*TILE_SIZE+TILE_SIZE/2+1;  //目的点 y 坐标，已经转换成中心点了
/*以上两行分别计算获得目的点的 x、y 坐标和英雄的当前位置坐标*/
    int hx = hero.x;                            //获得英雄的当前位置
    int hy = hero.y;
```

（2）确定了目的点的位置，下面就需要对英雄进行无级移动了，其移动的代码如下。

```
/*根据步骤（1）中计算出的步骤对英雄进行移动。每次移动时先判断位置和目的位置的大小关系，以此确定对英雄的当前位置做加法还是减法*/
```



```
for(int j=0;j<moves;j++){
    try{//先试一下
        Thread.sleep(HERO_MOVING_SLEEP_SPAN);}
    catch(Exception e){e.printStackTrace();}
    //计算英雄的 x 位移
    if(hx<destX){                                    //判断英雄坐标和目的点的大小
        hero.x = hx+j*HERO_MOVING_SPAN;              //修改英雄的 x 坐标
        hero.col = hero.x/TILE_SIZE;                 //及时更新英雄的行列值
        checkIfRollScreen(hero.direction);           //检查是否需要滚屏
    }else if(hx>destX){                              //判断英雄坐标和目的点的大小
        hero.x = hx-j*HERO_MOVING_SPAN;              //修改英雄的 x 坐标
        hero.col = hero.x/TILE_SIZE;                 //及时更新英雄的行列值
        checkIfRollScreen(hero.direction);           //检查是否需要滚屏
    }                                                //计算英雄的 y 位移
    if(hy<destY){                                    //判断英雄坐标和目的点的大小
        hero.y = hy+j*HERO_MOVING_SPAN;              //计算英雄的 y 坐标
        hero.row = hero.y/TILE_SIZE;                 //及时更新英雄的行列值
        checkIfRollScreen(hero.direction);           //检查是否需要滚屏
    }else if(hy>destY){                              //判断英雄坐标和目的点的大小
        hero.y = hy-j*HERO_MOVING_SPAN;              //修改英雄的 y 坐标
        hero.row = hero.y/TILE_SIZE;                 //及时更新英雄的行列值
        checkIfRollScreen(hero.direction);           //检查是否需要滚屏
    }}                                               //修正 x 和 y 坐标，修改英雄的占位格子
```

/*对英雄的 x、y 坐标以及其在地图中的行列值进行修正。这么做的原因是英雄的移动步径不能整除图元的大小，在执行完代码前一部分代码的无级移动后，英雄所处的位置可能不在目的格子的中心，因此需要进行修正*/

```
    hero.x = destX;                                  //修正英雄的 X 坐标
    hero.y = destY;                                  //修正英雄的 Y 坐标
    hero.col = destCol;                              //修改英雄在地图中的行数
    hero.row = destRow;                              //修改英雄在地图中的列数
    //修正 offsetX、y
```

/*GameView 的 offsetX 和 offsetY 进行修正的代码。其原因同样是因为英雄的移动步径不能整除图元大小。对 offsetX 和 offsetY 进行修正的原则是如果当前的偏移量小于英雄的移动步径（向左或向上滚屏后的结果），则舍去该偏移量。如果当前偏移量与地图图元的差值小于英雄移动步径时（向右或向下滚屏的结果），则将该偏移量进位，修改相应的 startRow 和 startCol 并将偏移量置零*/

```
    if(gv.offsetX<HERO_MOVING_SPAN){                 //舍去
        gv.offsetX = 0;
    }else if(gv.offsetX>TILE_SIZE - HERO_MOVING_SPAN){
    /*图元大小与 offsetX 的差小于英雄移动步进*/
    if(gv.startCol + GAME_VIEW_SCREEN_COLS < MAP_COLS -1){        //进位
        gv.offsetX=0;
        gv.startCol+=1;
    }}
if(gv.offsetY<HERO_MOVING_SPAN){//如果偏移量小于英雄步进
        gv.offsetY = 0;                              //舍去
    }else if(gv.offsetY>TILE_SIZE - HERO_MOVING_SPAN){
```

```
        //图元大小与 offsetY 的差小于英雄移动步进
         if(gv.startRow + GAME_VIEW_SCREEN_ROWS < MAP_ROWS -1){    //进位
              gv.startRow+=1;
              gv.offsetY = 0;
         }}
         hero.direction = checkIfTurn();//检查是否需要拐弯
    /*调用 checkIfTurn 方法对英雄所处的新位置进行转弯检查，检查的依据是判断相对于英雄当前方向
的正前方以及左右两边是否有路可走，如果有路则从可选路中随机选取一条并返回新道路的方向*/
```

（3）进行到步骤（2）为止，英雄已经按照掷出的骰子点数前进了相应的地图格子数，此时应该停止英雄的移动并检查英雄是否与地图中的可遇实体发生相遇，其代码如下。

```
    this.setMoving(false);                         //停止移动
    //通过设置 isMoving 标志位来暂停 HeroGoThread 线程的执行
    hero.setAnimationDirection(hero.direction%4);
    //将英雄的动画段设置为相应方向上的静止态
    //检查停留点有没有可遇的东西
    if(!checkIfMeet()){//如果没遇到，就设置状态待命，骰子继续变换
    /*调用 checkIfMeet 方法检查英雄是否遇到了地图中可遇实体，如果遇到可遇实体，该方法将进行相
应处理并返回 true，否则将返回 false*/
         this.gv.setStatus(0);                     //重新设置 GameView 的状态 0 为待命状态
         gv.gvt.setChanging(true);                 //继续让骰子变换
    //以上两行为 checkIfMeet 方法返回 false 时执行的操作
    }
    hero.growStrength(gv.currentSteps);            //增加英雄体力
```



说明

由于篇幅有限，上述步骤中调用的 checkIfRollScreen 和 checkIfMeet 方法的具体代码将不予列出。

任务 20　HeroBackDataThread 的开发

【任务情境】

HeroBackDataThread 是 Hero 类的另一个辅助线程。

【相关知识】

该线程的休眠时间比较长，其 run 方法中主要进行的工作如下。

☑　减少英雄及其已占领城池的粮草，如果某个城池发生粮草危机，则提示玩家进行



相应处理。

- ☑ 如果英雄目前有正在科研的项目，则推进项目的进度。如果项目已完成，则通过提示告知玩家。
- ☑ 随机选择一个城池向英雄占领的城池发动袭击，并提示玩家进行相应处理。
- ☑ 判断英雄的金钱、随身士兵、兵力和粮草是否超过一定值。如果超过了一定值，则产生一场沙尘暴对英雄的金钱、兵力和粮草进行一定衰减，并通过提示告知玩家。
- ☑ 线程长时间休眠。

上述为 HeroBackDataThread 的主要功能，由于篇幅所限，同时该类还涉及到后面要介绍的游戏提示模块的内容，故在此将不对其实现代码进行详细介绍。

【项目小结】

本项目主要对英雄角色模块的开发进行了介绍，该模块是游戏的实体模块之一，涉及的类有 Hero、HeroGoThread、HeroBackDataThread、General 和 Research，该部分详细讲解了其开发方法，类似游戏的开发过程与此大同小异，希望读者能够掌握。

📖 小知识十三：关于游戏角色设计

1. 游戏角色的定义

在实际游戏制作工作中，概念设计游戏原画要针对游戏的需要，或者根据自己对游戏所需角色的理解，根据游戏剧本的需要创作出游戏角色。创作出的游戏角色必须符合该游戏的世界观。

一名高水平的游戏原画设计师必须具备对各种题材游戏造型的准确理解和良好把握能力。现在的游戏类型与游戏题材多种多样，比较庞大的包括魔幻（魔法类，多数以龙与地下城或者基于此创造）、中国古典题材（中国武侠、历史题材的改编、中国古典原创的魔幻）、现代战争题材的改编等。作为一个游戏美术创作人员对以上题材，一定要有一个良好的把控能力。这就需要我们进行积累，并不断丰富自己，以便于在创作过程中得心应手。对于游戏美术的积累是多方面的，最重要的是量的积累：一方面是对于各种知识的积累，比如对文字类内容的阅读量，《龙枪》系列、《被遗忘的国度》系列都是基于龙与地下城世界观原创的小说，《三国演义》、金庸先生的武侠小说，都是被广为制作的题材，一名游戏原画设计师就像是一个编剧，对这些题材如果不了解是无法演绎好剧本中的角色的；另一方面就是绘画的技法和设计思路的积累，这对一名游戏原画设计师也是非常重要的。

2. 角色设计的总体认识

设计一个游戏中的角色，首先一定要明确认知所要设计的是什么。游戏是什么样的题材、是什么样的类型、是什么样的世界观；角色是什么样的性格、什么职业、什么种族、什么阵营、外貌、服饰、道具等。游戏原画是制定整个游戏的美术风格的重要环节，所以对游戏原画的要求也就更高。

（1）首先要根据制作团队初期对游戏的文字企划（或者策划的策划案），把握好游戏



的世界观，了解游戏类型与题材，明确游戏的制作思想。

（2）根据文字策划案制定游戏的美术风格，这可能是一个试验阶段或者说探讨阶段，总之最后由游戏特点来确定美术风格。风格一定要遵循游戏世界观。

（3）美术风格确定后进入场景与角色的创作设计阶段，每个角色、场景在游戏美术风格的大前提下去进行创作设计，要符合游戏中的要求。

3. 总体造型设计的注意事项

（1）在运用形、色、光等造型手段塑造形象时，要使人物间的服饰搭配、场景间的衔接转换协调统一。

（2）诸形象既要有个性色彩，又要构成整体基调；场景既要鲜明真切，又要以突出人物形象与性格特征为主，同时还要考虑采用的表现形式应与游戏的风格样式相适应。总体造型设计的关键是把握现象的总体感与表现形式的统一。

（3）整体造型设计还应照顾全局分清主次。在以景写情或刻画各单元场景时，要使景物情真意切，气氛符合要求；在重点表现人物时，又要让景物起烘托陪衬作用，以突出人物为前提。

角色在整体设计过程中需要考虑的因素很多，主要包括角色的基本画法、体态特征、头部特征、服装和性格特点等。

4. 游戏角色设计小技巧

角色设计就是一个决定在游戏中加入哪些角色的行为。就像一个完整的乐队一样，需要不同角色的完美配合、和谐共处，才能演奏出美妙的音乐。

在音乐中，有的乐器可能不适合与某些乐器合奏，如果强行合奏，不仅演绎不出美妙的音乐，还会影响整场表演的效果。同样地，在进行游戏角色设计时，有的角色不能将它们强扭到一块，而有的角色则应该将它们放到一块。

当把正面角色或反面角色组成一个团队时，需要确保队伍中所有的角色能很好地配合。你选择需要演奏的音乐曲目（原创或经典等类型，一般后者的风险比较小），而它们必须适合这样的曲目表演。正面角色与反面角色的对决需要能很好地表达音乐本身的魅力。

以下是几个进行游戏角色设计时的小技巧：

☑ 外形：黑白、胖瘦、大小……角色的视觉形象应该与故事情节匹配，否则就是失败的设计。

☑ 声音：凶神恶煞、雄浑强健、柔弱、女人味、很 man、很嗲、很诡异……角色的声音应该符合情节需要，让人听着自然。

☑ 装扮：对角色的整体形象而言，装扮起着很重要的作用，不可忽视。

☑ 动作：笨重、有力、轻快、出其不意……这都是在角色设计时需要考虑的。

当我们把音乐和角色设计联系在一起时，可以把音乐作为故事设计的灵感。一首好的音乐，可能会带来意想不到的创意。



综合实训九　微博随身之登录注册模块的实现

【问题情境】

本实训将介绍微博随身 Android 端各个功能模块的实现，本实训将介绍登录注册模块的实现，所涉及的类主要为 LoginActivity、RegActivity 及程序中的工具类 MyConnector 和 ConstantUtil。

【拓展知识】

1. 登录模块的开发

LoginActivity 是应用程序启动时首先被启动的 Activity，下面将介绍登录模块的开发，其步骤如下。

（1）开发 XML 布局文件，在项目 res\layout 目录下新建一个 login.xml，在其中输入如下代码。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
            android:orientation="vertical"
            android:background="@drawable/back"
            android:layout_width="fill_parent"<!--垂直分布的总线性布局-->
<!--显示账号输入的现象布局-->
<LinearLayoutandroid:orientation="horizontal"
            android:layout_gravity="center_horizontal"android:paddingTop="25px"
            android:layout_width="wrap_content"android:layout_height="wrap_content">
<TextView android:text="@string/tvUid"
            android:layout_width="100px" style="@style/text"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"/>
            <EditText android:id="@+id/etUid"android:singleLine="true"
            android:layout_width="150px"android:layout_height="wrap_content"/>
</LinearLayout>
<!--显示密码输入的线性布局-->
<LinearLayoutandroid:orientation="horizontal"android:layout_gravity="center_horizontal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
<TextView android:text="@string/tvPwd"
            android:layout_width="100px" style="@style/text"
            android:layout_height="wrap_content"android:layout_gravity="center_vertical" />
<EditText  android:id="@+id/etPwd"android:singleLine="true"
            android:password="true"
            android:layout_width="150px"android:layout_height="wrap_content"/>
</LinearLayout>
```

```
28 <!--显示 checkBox 和退出按钮的线性布局-->
29 <LinearLayoutandroid:orientation="horizontal"android:gravity="center_horizontal"
30 android:layout_width="wrap_content"android:layout_height="wrap_content>
31 <CheckBoxandroid:id="@+id/cbRemember"android:text="@string/cbRemember"
32         android:layout_gravity="center_horizontal"android:checked="false"
33         android:textColor="@color/character"android:layout_width="wrap_content"
34         android:layout_height="wrap_content"/>
35 <ImageButton android:id="@+id/ibExit" android:src="@drawable/exit"
36         android:layout_width="60px"android:layout_height="60px"/>
37 </LinearLayout>
38 <!-- 显示登录注册按钮的线性布局 -->
39 <LinearLayoutandroid:orientation="horizontal"android:layout_gravity="center_horizontal"
40         android:layout_width="wrap_content"android:layout_height="wrap_content>
41 <Button android:id="@+id/btnLogin"  style="@style/button" android:layout_width="120px"
42         android:layout_height="wrap_content" android:text="@string/btnLogin"/>
43 <Button    android:id="@+id/btnReg"style="@style/button" android:layout_width="120px"
44         android:layout_height="wrap_content"android:text="@string/btnReg"/>
45 </LinearLayout>
46 </LinearLayout>
```

☑ 第 2～5 行声明了垂直分布的线性布局，该布局中包括其他 4 个线性布局。

☑ 第 7～16 行声明了水平分布的线性布局，该布局包括提示输入账号的 TextView 和接收用户输入的 EditText 控件。

☑ 第 18～24 行声明了水平分布的线性布局，该布局中包括提示输入密码的 TextView 控件和接收用户输入密码的 EditText 控件。

☑ 第 26～34 行声明了一个水平分布的线性布局，该布局中包含一个用于记录用户账号和密码的 CheckBox 以及一个用于退出程序的 ImageButton。

☑ 第 36～42 行声明了一个水平分布的线性布局，该布局中包含登录和注册两个按钮控件。

（2）实现登录功能的 Activity 是 LoginActivity，其代码如下。

```
package wyf.wpf;                                   //声明包语句
import static wyf.wpf.ConstantUtil.*;              //静态引入 ConstantUtil 类的静态成员
…//此处省略部分引入相关类的代码
import android.widget.Toase;                       //引入相关类
public class LoginActivity extends Activity {
    MyConnector mc = null;                         //声明 MyConnector 对象引用
    ProgressDialog pd;                             //声明 ProgressDialog 对象引用
    @Override
    public void onCreate(Bundle savedInstanceState) { //重写 onCreate 方法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login);            //设置当前屏幕
        checkIfRemember();                         //检查是否曾经记录过账号和密码
        Button btnLogin = (Button)findViewById(R.id.btnLogin);  //获得登录 Button 对象
        btnLogin.setOnClickListener(new View.OnClickListener() {  //为“登录”按钮添加监听器
            public void onClick(View v) {          //重写 onClick 方法
```

```
16                      pd = ProgressDialog.show(LoginActivity.this, "请稍候", "正在连接服务
                        器...", true, true);
17                      login();                                //调用 login 方法获得
18                  }
19              });
20        Button btnReg = (Button)findViewById(R.id.btnReg);        //获得注册 Button 对象
21            btnReg.setOnClickListener(new View.OnClickListener() { //为“注册”按钮添加监听器
22                public void onClick(View v) {                      //重写 onClick 方法
23                    Intent intent = new Intent(LoginActivity.this, wyf.wpf.RegActivity.class);
24                    startActivity(intent);                         //启动负责注册的 Activity
25                    finish();                                      //结束本 Activity 的运行
26                }
27            });
28            ImageButton ibExit = (ImageButton)findViewById(R.id.ibExit);
            //获得退出 ImageButton 对象
29            ibExit.setOnClickListener(new View.OnClickListener() {  //为“退出”按钮添加监听器
30                public void onClick(View v) {
31                    android.os.Process.killProcess(android.os.Process.myPid());
                    //结束进程，退出程序
32                }
33            });
34        }
35        public void login(){                                       //方法：连接服务器进行登录
36        public void rememberMe(String uid,String pwd){//方法：将用户的 id 和密码存入 Preferences}
37        public void checkIfRemember(){                 //方法：从 Preferences 中读取用户名和密码
38        protected void onDestroy() {
39            if(mc != null){                            //判断 MyConnector 是否为 null
40                mc.sayBye();                           //调用通知服务器客户端结束通信
41                mc = null;
42            }
43            super.onDestroy();                         //调用父类的 onDestroy 方法
44        }
45    }
```

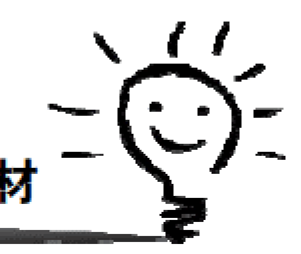
☑ 第 6 行声明了一个 MyConnector 对象的引用，该对象将负责与服务器通信。第 7 行声明了 ProgressDialog 对象的引用，该对象负责在连接网络时显示进度对话框。

☑ 第 12 行调用了 checkIfRemember 方法检查用户上次登录成功时是否保存了账号和密码。如果程序中已经保存了账号和密码，将会直接显示到账号和密码的文本框中。

☑ 第 13～19 行获得了屏幕中登录 Button 对象并为该对象添加监听器，第 22～26 行为重写的 onClick 方法，在该方法中首先创建一个 Intent 对象，然后调用 Activity 的 startActivity 方法启动 RegActivity，该 Activity 的功能是让用户注册新账号。

☑ 第 28～33 行获得屏幕中的 ImageButton 对象，并为该按钮添加监听器，第 30～32 行为重写的 onClick 方法，在该方法中调用 android.os.Process 类的 killProcess 方法结束进程退出程序。

☑ 第 35～37 行省略了部分成员方法的代码，将在后面的步骤中补全。



（3）在 LoginActivity.java 中使用到了 MyConnector 对象，该类对象的主要功能是与服务器进行通信，MyConnector 对象中包含与连接到服务器的 Socket，可以通过该 Socket 连接获得输入/输出流进行数据的发送和接收。MyConnector 类的代码如下。

```
1   package wyf.wpf;                                                    //声明包语句
2   import java.io.DataInputStream;                                     //引入相关类
3   import java.io.DataOutputStream;                                    //引入相关类
4   import java.net.Socket;                                             //引入相关类
5   public class MyConnector{
6       Socket socket = null;                                           //声明 Socket 对象
7       DataInputStream din = null;                                     //声明数据输入流对象
8       DataOutputStream dout = null;                                   //声明数据输出流对象
9       public MyConnector(String address,int port){
10          try{
11              socket = new Socket(address,port);                      //创建 Socket 连接
12              din = new DataInputStream(socket.getInputStream());     //获得输入流
13              dout = new DataOutputStream(socket.getOutputStream());//获得输出流
14          }catch(Exception e){
15              e.printStackTrace();                        //捕获打印异常
16          }
17      }
18      public void sayBye(){                               //方法：断开连接，释放资源
19          try{
20              dout.writeUTF("<#USER_LOGOUT#>");           //发出断开连接消息
21              din.close();                                //关闭输入流 DataInputSream
22              dout.close();                               //关闭输出流 DataOutputStream
23              socket.close();                             //关闭 Socket 连接
24              socket=null;
25          }catch(Exception e){
26              e.printStackTrace();                        //捕获并打印异常
27          }}}
```

☑ 第 6～8 行声明了与服务器进行通信的 Socket 连接，以及由 Socket 连接中获取的输入/输出流对象的引用，在代码第 9～17 行 MyConnector 的构造器中将对这些成员变量进行初始化。

☑ 第 18～27 行为 sayBye 方法，该方法的主要功能是向服务器发出客户端下线的通知，该方法应该放在每个使用 MyConnector 的 Activity 的 onDestroy 方法中执行，这样能够确保无用的 MyConnector 对象及时被关闭和释放。

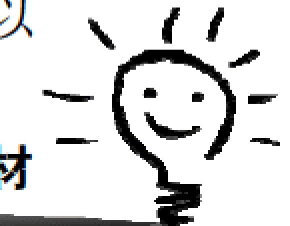
（4）在登录界面，用户单击“登录”按钮后会调用 login 方法与服务器通信进行登录验证，该方法的代码如下。

```
public void login(){                                //方法：连接服务器进行登录
    new Thread(){                                   //新创建一个进程
        public void run(){
```



```
                Looper.prepare();                    //开启一个消息循环
                    try{
                        if(mc == null){              //如果 MyConnector 为 null 则创建一个
                            mc = new MyConnector(SERVER_ADDRESS, SERVER_PORT);
                        }
                        EditText etUid = (EditText)findViewById(R.id.etUid);  //获得账号 EditText
                        EditText etPwd = (EditText)findViewById(R.id.etPwd);//获得密码 EditText
                        String uid = etUid.getEditableText().toString().trim();//获得输入的账号
                        String pwd = etPwd.getEditableText().toString().trim();//获得输入的密码
                        if(uid.equals(" ") || pwd.equals(" ")){      //判断输入是否为空
                    Toast.makeText(LoginActivity.this, "请输入账号或密码!",
                        Toast.LENGTH_SHORT).show();           //输出提示消息
                    return;
                    }
                String msg = "<#LOGIN#>"+uid+"|"+pwd;             //组织要返回的字符串
                mc.dout.writeUTF(msg);                            //发出消息
                String receivedMsg = mc.din.readUTF();            //读取服务器发来的消息
                pd.dismiss();
                if(receivedMsg.startsWith("<#LOGIN_SUCCESS#>")){
                //收到的消息为登录成功消息
                            receivedMsg = receivedMsg.substring(17);
                            String [] sa = receivedMsg.split("\\|");
                            CheckBox cb = (CheckBox)findViewById(R.id.cbRemember);
                            //获得 CheckBox 对象
                            if(cb.isChecked()){          //检查用户是否选择记录账号密码
                            rememberMe(uid,pwd);         //将用户名和密码记录下来
                            }
                Intent intent = new Intent(LoginActivity.this,FunctionTabActivity.class);
                intent.putExtra("uno", sa[0]);                  //向 Intent 对象设置 Extra
                startActivity(intent);                          //启动 FunctionTabActivity
                finish();                                       //结束本 Activity 的执行
            }
            else if(msg.startsWith("<#LOGIN_FAIL#>")){          //收到的消息为登录失败
                    Toast.makeText(LoginActivity.this, msg.substring(14),
                    Toast.LENGTH_LONG).show();
                    Looper.loop();                              //执行消息队列中的消息
                    Looper.myLooper().quit();                   //结束消息队列
                    }
            }catch(Exception e){
                        e.printStackTrace();                    //捕获并打印异常
                    }
            }
        }.start();
}
```

☑ 在 login 方法中创建并启动了一个独立的线程负责与服务器进行通信，这样做可以



让程序有更好的交互性和稳定性，建议读者在开发类似程序时将需要联网的代码放到单独的线程中执行。

☑ 第 6～8 行首先判断 MyConnector 对象是否为 null，如果为 null，则创建 MyConnector 对象，代码第 7 行使用到的 SERVER_ADDRESS 和 SERVER_PORT 常量来自于 ConstantUtil，该类存放了应用程序会使用到的常量，这样便于修改和维护。

☑ 第 9～17 行对用户输入的登录信息进行验证。首先获得账号和密码的文本框中的内容，判断其是否为空，如果为空，则提示用户重新输入账号和密码。

☑ 第 18～21 行代码将用户输入的账号密码组装起来，并加上消息头发送到服务器。当收到服务器端的登录反馈信息后，在代码第 21 行关闭进度对话框。

☑ 第 22～39 行对服务器返回的消息进行判断并执行不同的操作的代码。如果返回登录成功的消息，则检查用户是否选择了记录账号密码的选项，如果选择了该选项，就调用 rememberMe 方法记录账号密码，然后启动 FunctionTabActivity 进入个人中心。如果返回登录失败的消息，则向用户提示登录失败。

2. 记录账号密码功能的开发

在用户登录界面，用户可以选择让程序记住自己输入的账号和密码，这样如果登录成功，下次再启动该程序将直接显示上次输入的账号和密码。该功能通过 Android 平台下的 Preferences 来实现，实现该功能的 rememberMe 和 checkIfRemember 方法的代码如下。

```
1    public void rememberMe(String uid,String pwd){   //方法：将用户的 ID 和密码存入 Preferences
2       SharedPreferences sp = getPreferences(MODE_PRIVATE);  //获得 Preferences
3       SharedPreferences.Editor editor = sp.edit();        //获得 Editor
4       editor.putString("uid", uid);                       //将用户名存入 Preferences
5       editor.putString("pwd", pwd);                       //将密码存入 Preferences
6       editor.commit();                                    //提交修改
7       }
8    public void checkIfRemember(){                         //方法：从 Preferences 中读取用户名和密码
9       SharedPreferences sp = getPreferences(MODE_PRIVATE);  //获得 Preferences
10      String uid = sp.getString("uid", null);             //读取账号
11      String pwd = sp.getString("pwd", null);             //读取密码
12      if(uid != null && pwd!= null){                      //如果 Preferences 是否已存账号和密码
13              EditText etUid = (EditText)findViewById(R.id.etUid); //获得屏幕中的账号文本框控件
14              EditText etPwd = (EditText)findViewById(R.id.etPwd);//获得屏幕中的密码文本框控件
15              CheckBox cbRemember = (CheckBox)findViewById(R.id.cbRemember);
                //获得 CheckBox 控件
16              etUid.setText(uid);                         //将 Preferences 中存储的数据显示
17              etPwd.setText(pwd);
18              cbRemember.setChecked(true);                //设置 CheckBox 的选中状态
19          }
20   }
```

☑ 第 1～7 行为 rememberMe 方法的代码，该方法的功能是将指定的账号和密码存储到 Preferences 中。



☑ 第 8～20 行为 checkIfRemember 方法的代码，该方法的功能是读取 Preferences 中的数据，如果之前保存过账号和密码，则将显示到屏幕中相应的文本框中。

3. 服务器验证登录功能的实现

前面介绍了登录模块中 Android 部分的代码，当用户发送消息到达服务器时，服务器会创建 ServerAgent 线程负责与客户端通信，ServerAgent 类中处理用户登录请求的代码如下。

```
if(msg.startsWith("<#LOGIN#>")){                    //消息为登录
        String content = msg.substring(9);          //提取消息内容
        String [] sa = content.split("\\|");         //分离出账号和密码手段
        ArrayList<String> result = DBUtil.checkLogin(sa[0], sa[1]);
                                                    //调用 DBUtil 类的方法验证登录
                if(result.size()>1){                //登录成功
                    StringBuilder sb = new StringBuilder();
                    sb.append("<#LOGIN_SUCCESS#>"); //设置返回消息的头
                    for(String s:result){                  //添加用户登录成功信息
                        sb.append(s);
                        sb.append("|");                    //添加“|”
                    }
                    String loginInfo = sb.substring(0,sb.length()-1);
                    dout.writeUTF(loginInfo);              //返回用户的基本信息
        }
        else{                                              //登录失败
                    String loginInfo = "<#LOGIN_FAIL#>"+result.get(0);
                    //设置返回消息
                    dout.writeUTF(loginInfo);
                }
            }
```



说明

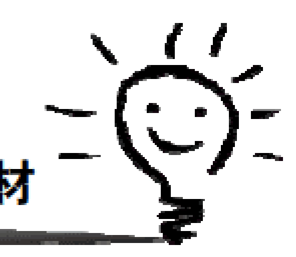
ServerAgent 在收到以“<#LOGIN#>”为头的消息后，会从中提取用户的账号和密码，并调用 DBUtil 类的 checkLogin 方法进行验证，然后根据验证结果向客户端发送不同的反馈消息。

4. 注册模块的开发

下面将介绍注册模块的开发。注册模块的功能主要由 RegActivity 来实现，下面将逐步介绍注册模块的实现。

（1）开发注册界面的布局，此布局由位于项目 res\layout 目录下的 reg.xml 来实现。该布局文件的代码比较长，由于篇幅有限只列出其大概框架。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```



```
        android:orientation="vertical" android:gravity="center_horizontal"
        android:background="@drawable/back"android:paddingTop="25px"
        android:layout_width="fill_parent"android:layout_height="fill_parent"><!--声明一个线性布局-->
    <LinearLayout android:orientation="horizontal"
        android:layout_width="wrap_content"android:layout_height="wrap_content">
    <!-- 声明一个水平分布的线性布局，用户在此输入昵称 -->
    </LinearLayout>
    <LinearLayout android:orientation="horizontal"
        android:layout_width="wrap_content"android:layout_height="wrap_content>
    <!-- 声明一个水平分布的线性布局，在此输入密码 -->
    </LinearLayout>
    <LinearLayout android:orientation="horizontal"
        android:layout_width="wrap_content"android:layout_height="wrap_content">
    <!-- 声明一个水平分布的线性布局，用户在此输入确认密码 -->
    </LinearLayout>
    <LinearLayout android:orientation="horizontal"
        android:layout_width="wrap_content"android:layout_height="wrap_content>
    <!--声明一个水平分布的线性布局，用户在此输入邮箱地址-->
    </LinearLayout>
    <LinearLayout android:orientation="horizontal"
        android:layout_width="wrap_content"android:layout_height="wrap_content">
    <!-- 声明一个水平分布的线性布局，用户在此输入心情 -->
    </LinearLayout>
    <LinearLayout android:orientation="horizontal"
        android:layout_width="wrap_content"android:layout_height="wrap_content">
            <!-- 声明一个水平分布的线性布局，用户在此单击注册或返回按钮 -->
    </LinearLayout>
    <LinearLayout android:orientation="vertical" android:visibility="gone" android:id="@+id/regResult"
        android:layout_width="wrap_content"android:layout_height="wrap_content">
    <!-- 声明一个水平分布的线性布局，该布局只在注册成功后显示用户获得的微博号 -->
    </LinearLayout>
    </LinearLayout>
```

说明

上述代码列出了注册界面的空间布局，主要是通过在垂直分布的线性布局中嵌套水平分布的线性布局来实现的。

（2）在登录界面如果不进行登录而是单击“注册”按钮，将会启动 RegActivity 进行注册，该 Activity 的代码如下。

```
package wyf.wpf;                    //声明包语句
import android.content.Intent;      //引入相关类
...//此处省略部分引入相关类的代码
import android.widget.Toast;        //引入相关类
```

```
5   public class RegActivity extends Activity{
6           MyConnector mc = null;                  //声明 MyConnector 对象
7           String uno = "";                        //记录用户的 ID
8           ProgressDialog pd= null;                //声明进度对话框
9           Handler myHandler = new Handler(){}     //创建 Handler 对象
10          protected void onCreate(Bundle savedInstanceState) {
11              super.onCreate(savedInstanceState);
12              setContentView(R.layout.reg);       //设置当前屏幕
13              Button btnReg = (Button)findViewById(R.id.btnReg); //获得“注册”按钮对象
14              btnReg.setOnClickListener(new View.OnClickListener() { //为“注册”按钮添加监听器
15              public void onClick(View v) {
16                  pd=ProgressDialog.show(RegActivity.this, "请稍候...", "正在连接服务器...",
                    false);
17                  register();
18              }
19          });
20          Button btnBack = (Button)findViewById(R.id.btnBack);    //获得“返回”按钮对象
21          btnBack.setOnClickListener(new View.OnClickListener() {//为“返回”按钮添加监听器
22              public void onClick(View v) {
23                  Intent intent = new Intent(RegActivity.this,LoginActivity.class);
                    //创建 Intent 对象
24                  startActivity(intent);          //启动 Activity
25                  finish();                       //结束本 Activity 的执行
26              }
27          });
28          Button btnEnter = (Button)findViewById(R.id.btnEnter); //获得“进入个人中心”按钮
29          btnEnter.setOnClickListener(new View.OnClickListener() {
            //为“进入个人中心”按钮添加监听器
30              public void onClick(View v) {
31                  Intent intent = new Intent (RegActivity.this,FunctionTabActivity.class);
                    //创建 Intent
32                  intent.putExtra("uno", uno); //设置 Extra 字段
33                  startActivity(intent);          //启动 FunctionTab
34                  finish();                       //关闭该 Activity
35              }
36          });
37      }
38      public void register(){                     //方法：连接服务器，进行注册
39      protected void onDestroy() {                //重写 onDestroy 方法，释放 MyConnector 对象
40  }
```

☑ 第 10～37 行为 RegActivity 的 onCreate 方法的代码，该方法的主要功能是设置当前的屏幕为按钮屏幕中的按钮添加监听器。

☑ 第 14～19 行为“注册”按钮添加了 OnClickListener 监听器，单击“注册”按钮后将显示进度对话框并调用 register 方法进行注册。

☑ 第 21～27 行为“返回”按钮添加了 OnClickListener 监听器，单击“返回”按钮将创建一个 Intent 对象启动 LoginActivity 并结束本 Activity 的运行。

☑ 第 29～36 行为“进入个人中心”按钮添加 OnClickListener 监听器，单击该按钮将创建 Intent 对象并启动 FunctionTabActivity 并结束本 Activity 的运行。

（3）上述代码中第 38 行省略了 register 方法的代码，该方法执行的操作为连接服务器进行注册，其代码如下。

```
public void register(){                    //方法：连接服务器，进行注册
    new Thread(){                          //创建线程
        public void run(){                 //重写 run 方法
            Looper.prepare();
            EditText etName = (EditText)findViewById(R.id.etName);  //获得昵称 EditText 对象
            EditText etPwd1 = (EditText)findViewById(R.id.etPwd1);  //获得密码 EditText 对象
            EditText etPwd2 = (EditText)findViewById(R.id.etPwd2);  //获得确认密码EditText对象
            EditText etEmail = (EditText)findViewById(R.id.etEmail);  //获得邮箱 EditText 对象
            EditText etStatus = (EditText)findViewById(R.id.etStatus);  //获得心情 EditText 对象
            String name = etName.getEditableText().toString().trim();     //获得昵称内容
            String pwd1 = etPwd1.getEditableText().toString().trim();     //获得密码内容
            String pwd2 = etPwd2.getEditableText().toString().trim();     //获得确认密码内容
            String email = etEmail.getEditableText().toString().trim();   //获得邮箱内容
            String status = etStatus.getEditableText().toString().trim(); //获得心情内容
                if(name.equals("") || pwd1.equals("") || pwd2.equals("") || email.equals("") ||
                 status.equals("")){                       //验证输入信息是否完整
                 Toast.makeText(RegActivity.this, "请将注册信息填写完整",
                                   Toast.LENGTH_LONG).show();
                return;
                }
                if(!pwd1.equals(pwd2)){                    //判断两次输入的密码是否一致
                    Toast.makeText(RegActivity.this, "两次输入的密码不一致！",
                                   Toast.LENGTH_LONG).show();
                    return;
                }
                try{
                    mc = new MyConnector(SERVER_ADDRESS, SERVER_PORT);
                    //创建 MyConnector
                    String regInfo = "<#REGISTER#>"+name+"|"+pwd1+"|"+email+"|"+ status;
                    mc.dout.writeUTF(regInfo);             //发出注册信息
                    String result = mc.din.readUTF();      //接收服务端反馈
                    pd.dismiss();                          //关闭进度对话框
                    if(result.startsWith("<#REG_SUCCESS#>")){//返回信息为注册成功
                        result= result.substring(15);      //去掉信息头
                        uno = result;                      //记录用户的 ID
                        myHandler.sendEmptyMessage(0);     //发出 Handler 消息
                        Toast.makeText(RegActivity.this, "注册成功！", Toast.LENGTH_
                        LONG).show();
```



```
37                              }
38                              else{        //注册失败
39                              Toast.makeText(RegActivity.this, "注册失败！请重试！", Toast.
                                LENGTH_LONG).show();
40                              }
41                          }catch(Exception e){   e.printStackTrace();}        //捕获并打印异常
42                      }
43                  }.start();
44              }
```

☑ 第 5～25 行代码执行的操作是获得用户输入的注册信息并对这些信息进行验证。

☑ 第 28 行将用户填写的注册信息包装为字符串并设置好消息头。第 29 行将组织好的注册信息发送到服务器。

☑ 第 32～40 行为对服务器反馈的消息进行判断的代码，如果注册成功，则从反馈消息中提取新用户的微博号记录到成员变量中，如果注册失败，则提示用户失败信息。

（4）上述代码中当用户注册成功时会在第 35 行发出一个 Handler 消息，该消息将通知 Activity 将布局文件中本来隐藏的控件显示出来。而 myHandler 对象的创建代码如下。

```
Handler myHandler = new Handler(){
        public void handleMessage(Message msg) {              //重写 handleMessage 方法
            switch(msg.what){                                 //对消息的 what 字段进行判断
            case 0:                                           //如果 what 值为 0
                ViewlinearLayout=(LinearLayout)findViewById(R.id.regResult);
                //获得隐藏的线性布局
                linearLayout.setVisibility(View.VISIBLE);     //设置可见性
                EditText etUno = (EditText)findViewById(R.id.etUno);
                //获得隐藏布局中的控件
                etUno.setText(uno);                           //显示新获得的微博号
                break;
            }
            super.handleMessage(msg);
        }
    };
```


说明

在 myHandler 对象的 handleMessage 方法中，当收到的消息的 what 字段为 0 时，就将 reg.xml 布局文件中原本隐藏的控件显示出来，并将新注册得到的微博号显示到 EditText 中。

📖 小知识十四：小窥验证码技术

目前，不少网站为了防止用户利用机器人自动注册、登录、灌水，都采用了验证码技术。所谓验证码，就是将一串随机产生的数字或符号，生成一幅图片，图片里加上一些干



扰像素（防止 OCR），由用户肉眼识别其中的验证码信息，输入表单提交网站验证，验证成功后才能使用某项功能。

很多朋友对验证码有疑问，各大论坛的用户也对验证码十分讨厌，觉得麻烦，下面我们来逐步解答一下：

最初的验证码，只是几个随机生成的数字。但是“道高一尺，魔高一丈”，很快就有能识别数字的软件了，“收藏家”们利用这种软件批量获取账号或是探测密码，因为软件可以不知疲倦地不断 submit。于是，出现了图片形式的验证码，还要加上无规则的背景，既然人眼都难以分辨，软件分辨起来就更有一定难度了。但是，腾讯开始采用汉字图片做验证码，可能意味着破解验证码的技术又有了新进展，带背景的数字或字母图片形式的验证码也可以被软件分辨了。

值得说明的是：验证码不同于注册码，注册码是软件作者根据提交的机器码通过特殊算法算出的，能让软件正常运行的密码。

1. 常见的验证码

（1）四位数字，随机的一数字字符串，最原始的验证码，验证作用几乎为零。

（2）CSDN 网站用户登录用的是 GIF 格式，目前常用的随机数字图片验证码。图片上的字符比较中规中矩，验证作用比上一个好。可惜，在 CSDN 使用它的第一天，破译它的程序就在论坛里发布了。

（3）腾讯网站用户登录用的验证码是 PNG 格式图片，图片用的随机数字+随机大写英文字母，整个构图有点张扬，每刷新一次，字符还会变换位置，有时候出来的图片，人眼都识别不了。

（4）MS 的 hotmail 申请时候的是 BMP 格式，随机数字+随机大写英文字母+随机干扰像素+随机位置。

（5）Google 的 Gmail 注册时候的是 JPG 格式，随机英文字母+随机颜色+随机位置+随机长度。

（6）其他各大论坛的是 XBM 格式，内容随机。

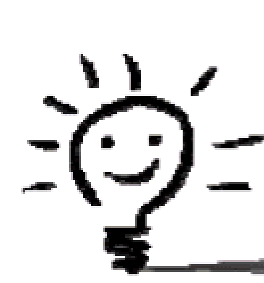
2. 验证码作用分析

验证码起源：因为攻击者会使用有害程序注册大量的 Web 服务账户（如 Passport）。攻击者可以使用这些账户为其他的用户制造麻烦，如发送垃圾邮件或通过同时反复登录多个账户来延缓服务的速度。在大多数情况下，自动注册程序不能识别此图片中的字符。简单地说，就是防止攻击者编写程序、自动注册、重复登录暴力破解密码。

验证码实现流程：服务器端随机生成验证码字符串，保存在内存中，并写入图片，发送给浏览器端显示，浏览器端输入验证码图片上的字符，然后提交服务器端，比较提交的字符和服务器端保存的字符是否一致。一致就继续，否则返回提示。就实际的效果来说，验证码只是增加了攻击的难度，而不可能完全防止。

（1）论坛中的验证码的作用

Web 站有时会碰到客户机恶意攻击，其中一种很常见的攻击手段就是身份欺骗它通过在客户端脚本写入一些代码，然后利用其客户机在网站论坛反复登录或者攻击者创建一个



HTML 窗体，其窗体如果包含了你注册窗体或发帖窗体等相同的字段，然后利用"http-post"传输数据到服务器，服务器会执行相应的创建账户、提交垃圾数据等操作，如果服务器本身不能有效验证并拒绝此非法操作，它会很严重耗费其系统资源，降低网站性能甚至使程序崩溃。

而现在流行的判断访问 Web 程序是合法用户的方式，就是采用一种叫“字符校验”的技术。Web 网站（像现在的动网论坛）采用的方法是为客户提供一个包含随机字符串的图片，用户必须读取这些字符串，然后登录窗体或者发帖窗体等用户创建的窗体一起提交即可。因为人眼很容易读出图片中的数字，但如果是一段客户端攻击代码，是很难识别验证码的，这样可以确保当前访问是来自一个人而非机器。

编程实现原理：使用某种动态编程语言，比如 PHP、ASP，随即生成一个随机数，大多为 4 位数字和字母或者是数字和字母的组合，生成以后用 GD 库的支持生成一张根据随机数来确定的图片，把随机数写入到 session 中，传递到要验证的页面，生成的图片显示给登录者，并要求登录者输入该随机数内容，提交到验证页面，验证 session 的内容和提交的内容是否一致，这就是大致的思路。那么如何编写验证码程序？相信 Google 一下，就有很多现成的代码。

（2）申请 QQ 号时输入验证码的作用

如今你要申请一个 QQ 号，需要输入很复杂的验证码：由若干个汉字组成，还有非常复杂的背景，使得有些汉字实在难以辨认。腾讯这么做，是为了防止有人利用软件批量获取 QQ 号码。每次提交都要输入随机生成的验证码，这是软件难以做到的。

3. 图片验证码技术之一：利用 XBM 格式图片

生成验证代码的技术有很多，这里只说与我们论坛有关系的这项技术。

x-xbitmap 格式图片（以下简称为 XBM 格式）的特殊性就在于它并不与 GIF、JPG 等图片格式一样，是一个真正的纯二进制图片格式，而是 ASCII 码文件是一个纯文本文件，在 Windows 系统下，系统浏览器会将它翻译成图片来进行显示。

4. 为什么要打补丁才能正常显示呢

在 WindowsXP SP2 更改后的安全策略中，基于安全因素的考虑，默认去掉了对 image/x-xbitmap 图片格式的支持（该图片的后缀名为.xbm）。为什么微软在 XP 的 SP2 升级包中又要禁止掉它呢？这是因为 XBM 的漏洞。

Microsoft Internet Explorer 和 Outlook Express 在处理 Web 页、HTML 邮件、E-mail 附件中畸形的 XBM 图像文件会导致系统崩溃，原因在于对 XBM 文件中的内容缺少检查，MSIE 按照图像规定的长度和宽度分配内存，攻击者可以提高超大的长度和宽度数值，从而导致系统消耗内存或者访问冲突。

换句话说，如果构造一个尺寸特别大的 XBM 文件，很容易导致 Windows 的内存耗尽，程序无响应或者死机。从它本身来说，这不算一个特别严重的漏洞，因为根据安全公告，无法造成溢出，不会存在太大的权限漏洞。但是由于 XP 的 SP2 强调安全性，因此将 XBM 功能禁用了。从这点上可以看出，SP2 对于安全的确比较重视，将有漏洞的功能基本上都补上或禁用了。

由此看来，以后我们访问某些采用生成 XBM 作为验证代码的站点时，就相当不方便了，如果有必要，可以通过简单的操作注册表恢复我们需要的功能。

解禁方法：

打开注册表（执行开始→运行→regedi 命令），然后进到键值[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet EXPlorer\Security]，将 blockXbm 的值改为 00000000（dword，双字节），没有该键值的话新建一个就可以了。

之后重新启用 IE 或者重新启动机器，XBM 格式的图片就可以看到了。

5. XBM 的趋势

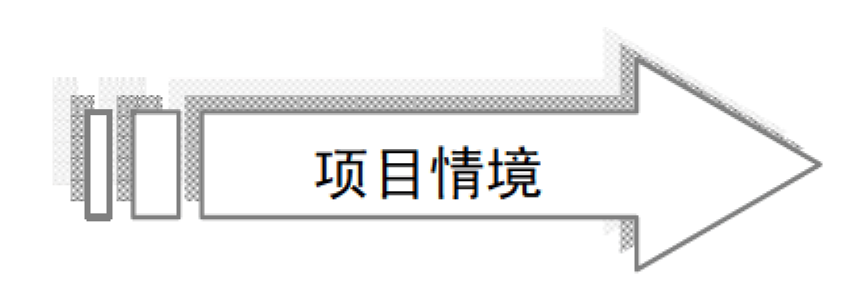
从 SP2 禁止 XBM 的趋势看出，微软似乎已经开始打算放弃对 XBM 格式的支持了。作为程序编写者，有必要未雨绸缪，寻找其他生成验证码的途径。在 PHP 中，可以通过调用 GD 库等方式生成 JPG/GIF 等图形格式的注册验证码，那么在 ASP 中有其他的办法么？

事实上图片验证密码的关键是不能在客户端留下图片的真实 URL，或可对应反推源地址的信息，因此 ASP 可以采用以下两种方式实现支持 SP2 的图形验证码。

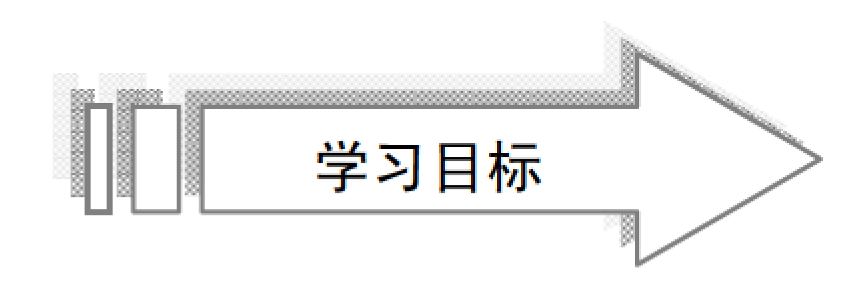
如果是购买的虚拟主机，那么可以采用将 JPG/GIF 图片放到数据库，然后用 session 传值的方式，最后利用 ASP 直接从数据库中输出图片，这种方法的好处是不需要特别设置服务器端，坏处则是每次生成验证图片时都会需要与数据库连接，增加了开销。

如果是有管理员控制权限的用户，可以考虑采用第三方组件来实现。个人推荐 ASP 图像组件 shotgraph，它的免费版本对生成的图形有一定限制，不过已经足够用来制作验证码了。

项目十
表示层界面模块的开发



前面已经介绍了英雄模块的相关类，本项目将介绍前台表示层的各个界面，主要包括各个 View 以及一些 View 辅助线程。



- 掌握竹简滚屏技术的框架代码、滚屏技术后台线程的开发方法
- 掌握 GameView 框架的搭建、成员变量的成员方法
- 掌握游戏界面绘制方法的框架、游戏主界面的绘制方法
- 掌握屏幕事件监听方法，使玩家能够与游戏进行交互
- 掌握定时读取 GameView 状态的方法，根据需要产生动画或缩略地图的滑入效果

任务 21　ScreenRollView 类的开发

【任务情境】

ScreenRollView 类采用竹筒滚屏技术，是在游戏菜单界面单击“初入江湖”后出现的游戏背景概述界面。

【相关知识】

其框架代码如下。

```
package wyf.ytl;
import android.content.res.Resources;                    //引入相关类
import android.graphics.Bitmap;                          //引入相关类
import android.graphics.BitmapFactory;                   //引入相关类
import android.graphics.Canvas;                          //引入相关类
import android.graphics.Color;                           //引入相关类
import android.graphics.Paint;                           //引入相关类
import android.view.MotionEvent;                         //引入相关类
import android.view.SurfaceHolder;                       //引入相关类
import android.view.SurfaceView;                         //引入相关类
import android.widget.Toast;                             //引入相关类
public class ScreenRollView extends SurfaceView implements SurfaceHolder.Callback{
/*成员变量的声明，主要是对界面中各个图片的坐标进行声明，这样在生成动画时只需要改变相应的
坐标值即可*/
    HDZGActivity activity;
    Bitmap bmpScroll;                                    //竹筒图片
    Bitmap bmpBack;                                      //背景图片
    DrawThread drawThread;                               //后台的重绘线程
    ScreenRollThread screenRollThread;                   //竹筒滚屏的后台线程
    float textSize = 23f;                                //字体的大小
    int scrollStartX = 320;                              //竹筒开始的 x 坐标
    int scrollStartY = 60;                               //竹筒开始的 y 坐标
    int characterEachLine = 10;                          //每行显示的汉字个数
    int characterSpanX = 36;                             //文字 x 方向间距
    int characterSpanY = 25;                             //文字 y 方向间距
    int characterNumber = 1;                             //已经显示了的字数
    int textStartX=237;                                  //文字起始的 x 坐标
    int textStartY=135;                                  //文字起始的 y 坐标
    int startChar=0;                                     //从字符串中的哪里开始输出，用于滚屏使用
```



```
        int maxChar = 60;                             //竹筒上最大能显示的文字个数
        int alpha = 255;                              //透明度
        int status;                                   //状态位，1 表示竹筒滑入，2 表示文字显示
    String msg = "东周末年，周室已经名存实亡，天下只是在名义上归于一家。各个诸侯国之间的战争
早已是烽火连天，各种吞并"+"和结盟屡见不鲜。不知你能否在这乱世靠着自己的才干谋略为自己守住一
方土地，甚至是和其他诸侯国一样，逐鹿"+"中原，并吞八荒？";
    //声明了一个描述游戏背景的字符串，当绘制文字时，从该字符串中截取
    /*该类的构造器，在构造器中初始化绘制线程以及动画教程，并对需要的图片资源进行初始化*/
    public ScreenRollView(HDZGActivity activity) {            //构造器
            super(activity);
            this.activity = activity;                     //得到 activity 的引用
            getHolder().addCallback(this);
            drawThread = new DrawThread(getHolder());     //初始化绘制线程
            screenRollThread = new ScreenRollThread(this);  //初始化动画线程
            initBitmap(getResources());                   //初始化图片资源
            status = 1;                                   //设置状态到竹筒滑入
        }
        //初始化图片
    public void initBitmap(Resources r){
            bmpScroll = BitmapFactory.decodeResource(r, R.drawable.scroll);//竹筒图片
        }
        //屏幕的绘制方法
    /*重写的 onDraw 绘制方法，该方法负责绘制屏幕中的信息*/
    public void onDraw(Canvas canvas){
            …//该处省略了界面的绘制代码，将在后面给出
    }
    /*3 个接口中方法的实现以及重绘线程实现与之前介绍的完全相同，在此就不再赘述*/
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
        }
    public void surfaceCreated(SurfaceHolder holder) {
            …//该处省略了绘制线程与动画线程的启动工作
        }
    public void surfaceDestroyed(SurfaceHolder holder) {
            drawThread.setFlag(false);
        }
    //后台的重绘线程
    class DrawThread extends Thread{                          //刷帧线程
        …//该处省略了绘制线程中的绘制代码
    } }
    */
```

任务 22　ScreenRollThread 线程类的开发

【任务情境】

前面已经对竹筒滚屏的界面进行了介绍，下面将对其后台线程进行开发，使竹筒滚动起来。

【相关知识】

其代码如下。

```
package wyf.ytl;
public class ScreenRollThread extends Thread{      //动画生成线程
    boolean flag;                                  //循环控制变量
    int sleepSpan = 100;
    ScreenRollView screenRoll;                     //声明 screenRoll 的引用
    int characterControl;                          //控制文字显示速度变量
    int charNumber;                                //记录显示文字的个数，包括显示了后备滚屏掩盖的

    public ScreenRollThread(ScreenRollView screenRoll){   //构造器
        super.setName("==ScreenRollThread");
        this.screenRoll = screenRoll;                      //得到 screenRoll 的引用
        flag = true;                                       //设置循环标志位
    }
    public void run(){                                     //线程执行方法
        while(flag){                                       //循环标志位为真时
            switch(screenRoll.status){
/*移动竹筒，当竹筒移动到位后，将界面状态设置成 2，进入文字显示阶段*/
            case 1://竹筒滑入
                screenRoll.scrollStartX -= 5;
                if(screenRoll.scrollStartX == 20){
                    screenRoll.status = 2;                 //进入文字显示阶段，停住不走了
                }
                break;
/*显示文字阶段，每次多出现一个文字，而当文字全部显示完时，将状态设成 3*/
            case 2://文字显示
                characterControl ++;
                if(characterControl == 3){
                    screenRoll.characterNumber ++;   //每次显示的文字都多一个
                    characterControl =0;
                    charNumber ++;
                    if(charNumber == screenRoll.msg.length()){  //显示结束
```

```
                    screenRoll.status = 3;          //3 是待命，一会要在这儿发 Handler
                }
            }
            break;
```

/*状态为 3 时的操作代码，每次循环将透明度降低 5 点，而当透明度降到 0 时，将状态值设成 4，并将 activity 发送 Handler 消息*/

```
        case 3://渐隐状态
            screenRoll.alpha -=5;             //改变透明度
            if(screenRoll.alpha == 0){        //当透明度到达 0 时
                screenRoll.status = 4;        //切屏
                screenRoll.activity.myHandler.sendEmptyMessage(11);
                this.flag = false;            //停止该线程
            }
            break;
        }
        try{
            Thread.sleep(sleepSpan);          //睡眠指定毫秒数
        }catch(Exception e){                  //捕获异常
            e.printStackTrace();              //打印异常信息
        }}}}
```

提示

因为篇幅有限，菜单（MenuView）、加载（LoadingView）、帮助（HelpView）和战斗（BattleField）等界面实现都非常简单，与之前介绍过的界面基本相同，在此不再介绍。

任务 23　游戏界面 GameView 的框架

【任务情境】

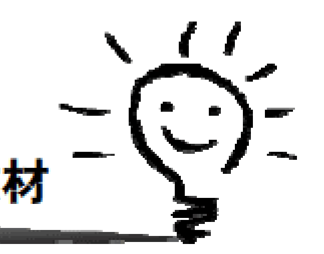
接下来将介绍该游戏最主要的界面——游戏界面 GameView，该界面绘制了游戏过程中的所有游戏信息。

【相关知识】

其开发步骤如下。

（1）首先搭建 GameView 的框架，其代码如下。

```
package wyf.ytl;
import static wyt.ytl.ConstantUtil.CITY_INFO.*;          //静态引入 ConstantUtil 的成员变量
…//该处省略了部分类的引入代码
```



```
import android.view.View;//引入相关类
public class GameView extends SurfaceView implements SurfaceHolder.Callback,View.OnTouchListener {
    ...//该处省略了成员变量的声明，将在后面给出
}
public GameView(HDZGActivity activity) {                //构造器
        ...//该处省略了构造器中对各种资源的初始方法
}
...//该处省略了声音的初始化和播放两个方法
/**
/*初始化所有用到的类*/
    public void initClass(){//初始化所有用到的类
        manPanelView = new ManPanelView(this);
        ...//该处省略了部分相关类的初始化工作
        tianXiaView = new TianXiaView(this);//初始化天下局势界面
    }
/*初始化地图信息及图片资源，初始化地图资源时，需要从 LayerList 的一个对象中读取地图信息*/
public void initMap(){//初始化地图
...//该处省略了地图资源的初始化工作
}
public static void initBitmap(Resources r){             //初始化图片资源的方法
...//该处省略了图片资源的初始化工作
}
/*各种信息的绘制方法，除了 onDraw 比较复杂外，其他的方法都只需将需要的图片绘制到指定位置
即可，而 onDraw 方法将在后面进行介绍*/
public void onDraw(Canvas canvas){                      //绘制屏幕
...//该处省略了屏幕的绘制方法
}
public void drawMiniMap(Canvas canvas,int col,int row){
...//该处省略了缩略地图的绘制方法
}
public void drawDice(Canvas canvas){                    //画骰子
...//该处省略了绘制骰子的工作
}
public void drawHeroData(Canvas canvas){                //主要的游戏数据
...//该处省略了游戏界面下面各种数据
}
*/
public int[] getDigitSequence(int data){                //获得一项数据的六位数码管显示序列
...//该处省略了游戏界面下面各种数据的数码管的绘制
}
public void drawDigits(Canvas canvas,int [] sequence,int x,int y){
//按照数码管数字序列画出 6 个数字
        for(int i=0;i<6;i++){
            canvas.drawBitmap(bmpDigit[sequence[i]], x+i*DIGIT_SPAN, y, null);
        }
    }
```

```
public void setCurrentDrawable(MyMeetableDrawable currentDrawable) {
            this.currentDrawable = currentDrawable;
    }
public void setStatus(int status){                                    //设置 GameView 的状态
            this.status = status;
    }
/*屏幕事件监听方法，根据玩家单击屏幕的事件进行相应的处理*/
public boolean onTouch(View view, MotionEvent event){                 //重写的监听器实现方法
    …//该处省略了事件监听方法的实现代码
}
/*这里方法的作用是将游戏中的线程以及声音停止，该方法会在游戏退出时被调用*/
public void stopGame(){
            …//该处省略了游戏界面声音以及线程的释放工作
    }
/*获得和设置当前的 GameAlert 的方法，Alert 为游戏过程中弹出的提示*/
    public GameAlert getCurrentGameAlert() {                          //返回当前的 Alert
            return currentGameAlert;
    }
    public void setCurrentGameAlert(GameAlert currentGameAlert) {     //设置当前的 Alert
            this.currentGameAlert = currentGameAlert;
    }
/*与前面介绍的各个 View 中的完全相同，在此不再赘述*/
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
    }
    public void surfaceCreated(SurfaceHolder holder) {                //当 View 被创建时被调用
    …//该处省略了相关线程的启动工作
    }
    public void surfaceDestroyed(SurfaceHolder holder) {              //当 View 被摧毁时被调用
            this.drawThread.setIsViewOn(false);
    }
    class DrawThread extends Thread{                                  //刷帧线程
    …//该处省略了线程中绘制代码
}}
*/
```

（2）下面给出 GameView 类中成员变量声明的代码。

```
private int status = 0;
/*绘制时的状态，0 正常游戏，1 英雄在走动，2 主菜单出现，3 选择武将，4 战斗动画，5 穷死结束
游戏，6 统一中国，100 人物属性界面，99 武将情报，98 使用计谋，97 城池管理，96 天下局势*/
    HDZGActivity activity;                        //activity 的引用
    DrawThread drawThread;                        //刷帧的线程
    Hero hero;                                    //英雄对象
    static Bitmap dialogBack;                     //存放对话框背景
    static Bitmap dialogButton;                   //存放对话框按钮
    static Bitmap [][] heroAnimationSegments;     //存放英雄所有动画段的图片
```

```
static Bitmap [] bmpDice;                                    //存放骰子的图片数组
static Bitmap dashboardBitmap;                               //下面用于管理的图片
static Bitmap darkBitmap;                                    //黑色外框的效果图
static Bitmap[] smallGameMenuOptions = new Bitmap[6];        //存放主菜单分割以后的图片
static Bitmap[] bmpDigit;                                    //存放数码管数字的图片
static Bitmap bmpBattleFiled;                                //战场背景图片
static Bitmap[] bmpHero;            //存放英雄的图片，分为静态和动态，代表英雄静止还是走路
int [][] notInMatrix=new int[MAP_ROWS][MAP_COLS];            //存放整个大地图的不可通过矩阵
int miniMapStartX = MINI_MAP_START_X;                        //初始情况下缩略地图开始的x坐标
int miniMapStartY = -160;                                    //初始情况下缩略地图开始的y坐标
boolean showMiniMap;                                         //是否显示迷你地图
int startRow = 0;                                            //屏幕在大地图中的行数
int startCol = 0;                                            //屏幕在大地图中的列数
int offsetX = 0;                    //屏幕定位点在大地图上的 x 方向偏移，用来实现无级滚屏
int offsetY = 0;                    //屏幕定位点在大地图上的 y 方向偏移，用来实现无级滚屏
LayerList layerList;                               //所有的层
MyMeetableDrawable [][] meetableMatrix;            //存放大地图的可遇矩阵
MyMeetableDrawable currentDrawable;                //记录当前碰到的可遇 Drawable 对象引用
MyMeetableDrawable previousDrawable;               //记录上一个碰到的可遇 Drawable 对象引用
MeetableLayer meetableChecker;
GameViewThread gvt;                                //后台修改数据的线程
BattleField battleField;
int suiXinBu=0;                                    //记录英雄施放随心步时选择的步数
General fightingGeneral = null;                    //存放当前出战的将军
ArrayList<Skill> skillToLearn;                     //将要学习的技能
int diceCount=2;                    //起作用的骰子的个数，学习了骑术会增加，最大到 3
int []diceValue = {1,3,5}; //存放骰子的值，其实是骰子图片数组的下标，0～5 代表 1～6 的骰子图片
int currentSteps;                                  //记录本次掷骰子需要走几步
ManPanelView manPanelView;                         //人物属性窗口
WuJiangView wuJiangView;                           //武将情报窗口
UseSkillView useSkillView;                         //使用计谋窗口
CityManageView cityManageView;                     //城池管理窗口
SelectGeneral selectGeneral;                       //选中出征武将窗口
TianXiaView tianXiaView;                           //天下局势窗口
GameAlert currentGameAlert;                        //记录当前的消息提示
ArrayList<CityDrawable> allCityDrawable = new ArrayList<CityDrawable>();  //存放所有敌方的城池
ArrayList<General> freeGeneral;                    //自由的将领，即不属于我方和敌方的
Paint paint;                                       //画笔
public static Resources resources;                 //声明资源对象引用
MediaPlayer mMediaPlayer;
SoundPool soundPool;                               //声音
HashMap<Integer, Integer> soundPoolMap;
```




说明

代码中各个成员变量的含义可见后面的注释，在对图片资源进行声明时，同样需要将其声明成静态变量。

任务 24　游戏界面绘制方法 onDraw

【任务情境】

任务 23 中介绍了 GameView 的框架，本任务将对其进行完善。

【相关知识】

首先是对 onDraw 绘制方法的完善，该方法主要负责界面的绘制工作，完善步骤如下。

（1）该方法的框架如下，根据游戏状态值的不同绘制不同的界面。

```
public void onDraw(Canvas canvas){
        //画的内容是 z 轴的，后画的会覆盖前面画的
        if(status == 0 || status == 1 || status == 2){          //游戏主界面时
            ...//该处省略了游戏主界面的绘制工作，代码将在后面给出
        }
        else if(status == 4){                                   //绘制战斗动面
            battleField.onDraw(canvas);                         //调用战斗动画的绘制方法
        }
        else if(status == 100){                                 //人物属性界面
            manPanelView.onDraw(canvas);                        //调用人物属性界面的绘制方法
        }
        else if(status == 99){                                  //绘制武将情报
            wuJiangView.onDraw(canvas);                         //调用武将情报界面的绘制方法
        }
        else if(status == 98){                                  //绘制使用计谋
            useSkillView.onDraw(canvas);                        //调用使用技能界面的绘制方法
        }
        else if(status == 97){                                  //绘制城池管理
            cityManageView.onDraw(canvas);                      //调用城池管理界面的绘制方法
        }
        else if(status == 3){                                   //绘制选择武将
            selectGeneral.onDraw(canvas);                       //调用选择武将界面的绘制方法
        }
```

```
        else if(status == 96){                  //绘制天下局势
            tianXiaView.onDraw(canvas);         //调用天下局势界面的绘制方法
        }
        paint.paint = new Paint();              //初始化画笔
        paint.setColor(Color.RED);              //设置颜色
        paint.setTextSize(38);                  //设置字体大小
    }
```

说明

该方法是对游戏的状态值进行判断，根据不同的状态绘制不同的信息，当游戏状态值表示不需要绘制游戏主界面时（即其值不为 0、1、2 时），调用相应封装类的绘制方法即可。

（2）游戏主界面的绘制，代码如下，将下列代码插入到第（1）步的代码第 3 行处。

```
/*清屏并复制与绘制相关的数据*/
    canvas.drawColor(Color.BLACK);      //清屏
    int heroX=hero.x;                   //英雄中心点的 x 坐标
    int heroY=hero.y;                   //英雄中心点的 y 坐标
    int hCol = heroX/TILE_SIZE;         //计算英雄中心点位于哪个格子
    int hRow = heroY/TILE_SIZE;         //计算英雄中心点位于哪个格子
    int tempStartRow = this.startRow;   //记录本次绘制时的定位行
    int tempStartCol = this.startCol;   //记录本次绘制时的定位列
    int tempOffsetX = this.offsetX;     //记录本次绘制时的 x 偏移
    int tempOffsetY = this.offsetY;     //记录本次绘制时的 y 偏移
/*绘制底层地图，从 LayerList 中得到底层地图的引用 1，然后对其表示地图信息的数组 mapMatrix
循环，得到其中的实体元素，再调用实体元素自身的绘制方法绘制地图*/
    for(int i=-1; i<=GAME_VIEW_SCREEN_ROWS; i++){          //绘制底层
        if(tempStartRow+i < 0 || tempStartRow+i>MAP_ROWS){//如果多画的那一行不存在，就继续
            continue;
        }
        for(int j=-1; j<=GAME_VIEW_SCREEN_COLS; j++){
            if(tempStartCol+j <0 || tempStartCol+j>MAP_COLS){//如果多画的那一列不存在，就继续
                continue;
            }
            Layer l = (Layer)layerList.layers.get(0);          //获得底层的图层
            MyDrawable[][] mapMatrix=l.getMapMatrix();
            if(mapMatrix[i+tempStartRow][j+tempStartCol] != null){
              mapMatrix[i+tempStartRow][j+tempStartCol].drawSelf(canvas,i,j,tempOffsetX,tempOffsetY);
            }
        }
    }
```

/*绘制上层地图的信息，同样从 LayerList 得到上层的 Layer，然后循环绘制界面，当英雄绘制点时，调用英雄自身的绘制方法绘制英雄*/

```
        /*绘制上层*/
        for(int i=-1; i<=GAME_VIEW_SCREEN_ROWS; i++){
            if(tempStartRow+i < 0 || tempStartRow+i>MAP_ROWS){//如果多画的那一行不存在，就继续
                continue;
            }
            for(int j=-1; j<=GAME_VIEW_SCREEN_COLS; j++){
                if(tempStartCol+j <0 || tempStartCol+j>MAP_COLS){//如果多画的那一列不存在，就继续
                    continue;
                }
                Layer l = (Layer)layerList.layers.get(1);//获得上层的图层
                MyDrawable[][] mapMatrix=l.getMapMatrix();
                if(mapMatrix[i+tempStartRow][j+tempStartCol] != null){
                mapMatrix[i+tempStartRow][j+tempStartCol].drawSelf(canvas,i,j,tempOffsetX,tempOffsetY);
                }
                if(hRow-tempStartRow == i && hCol-tempStartCol == j){          //英雄在这里
                hero.drawSelf(canvas,tempStartRow,tempStartCol,tempOffsetX,tempOffsetY);
                }
            }
        }
```

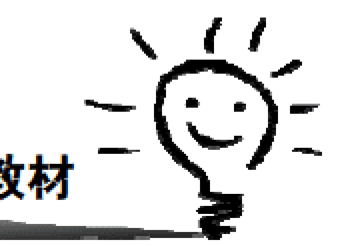
/*绘制游戏界面的下方仪表盘，首先绘制背景图片，然后在指定位置绘制骰子、英雄个人数据以及英雄的头像*/

```
        canvas.drawBitmap(dashboardBitmap, 0, ConstantUtil.SCREEN_HEIGHT-dashboardBitmap. getHeight(),
paint);                                     //绘制仪表盘
        drawDice(canvas);                   //绘制骰子
        drawHeroData(canvas);               //绘制仪表盘上的数据
        canvas.drawBitmap(bmpHero[status==1?1:0], 0, HERO_FACE_START_Y, null);//画英雄的头像
    /*先判断是否需要绘制缩略地图，需要时调用 drawMiniMap 方法绘制缩略地图*/
    if(showMiniMap){                        //检查是否需要显示小地图
            drawMiniMap(canvas,hCol,hRow);
        }
```

任务 25　游戏界面屏幕监听方法 onTouch

【任务情境】

任务 24 中介绍了游戏界面的绘制方法，但是玩家还是不能操控游戏，本任务将对屏幕事件监听方法进行介绍，使玩家能够与游戏进行交互。



【相关知识】

（1）onTouch 方法框架的搭建，其框架代码如下。

```
public boolean onTouch(View view, MotionEvent event) {          //屏幕事件监听
    if(event.getAction() == MotionEvent.ACTION_DOWN){          //捕捉屏幕被按下的事件
        int x = (int)event.getX();                              //得到 x 坐标
        int y = (int)event.getY();                              //得到 y 坐标
        /*游戏主界面中*/
        if(this.status == 0){
       ...//该处省略了游戏主界面显示时的屏幕事件处理代码，将在后面给出
    }else if(this.status == 2){                                 //主菜单出现时
      ...//该处省略了主菜单界面出现时的屏幕事件处理代码
        }else if(status == 100){                                //人物属性界面
            manPanelView.onTouchEvent(event);
        }
        else if(status == 99){                                  //武将情报界面
            wuJiangView.onTouchEvent(event);
        }
        else if(status == 98){                                  //使用计谋界面
            useSkillView.onTouchEvent(event);
        }
        else if(status == 97){                                  //城池管理界面
            cityManageView.onTouchEvent(event);
        }
        else if(status == 3){                                   //选择武将界面
            selectGeneral.onTouchEvent(event);
        }
        else if(status == 96){                                  //天下局势
            tianXiaView.onTouchEvent(event);
        }
    }
    return true;
}
```

说明

在该方法中，根据当前游戏状态的不同，即用户界面显示的不同，做出不同的监听。当用户界面正在显示主菜单时，根据单击的菜单按钮的不同做出不同的操作。而当游戏界面正在显示控制面板界面时，直接调用控制面板封装类中的事件处理方法即可，其中控制面板类将在项目十一中进行介绍。

（2）游戏界面正在显示游戏过程时的屏幕监听，即当游戏状态值为 0 时的处理代码如下。

/*根据玩家单击的位置判断所单击的是哪个按钮，然后根据单击按钮的不同初始化不同的数据，最后改变状态值



```
if(x>282 && x<310   && y>400 && y<470){       //单击主菜单
    darkBitmap = BitmapFactory.decodeResource(getResources(), R.drawable.dark);
    this.status = 2;
}else if(x>62 && x<80 && y>408 && y<426){     //单击了人物属性按钮
    this.manPanelView.initData();
    this.status = 100;
}else if(x>62 && x<80 && y>427 && y<445){     //武将情报
    this.wuJiangView.initData();
    this.status = 99;
}else if(x>42 && x<61 && y>427 && y<445){     //城池管理
    this.cityManageView.initData();
    this.status = 97;
}else if(x>22 && x<41 && y>427 && y<445){     //天下局势
    this.tianXiaView.initData();
    this.status = 96;
}else if(x>2 && x<21 && y>427 && y<445){      //使用计谋
    this.useSkillView.initData();
    this.status = 98;
}
```

/*单击人物头像时的处理代码。先判断状态值是否为 0，为 0 表示英雄正在待命状态，可以掷骰子走路。然后掷骰子并根据所掷子的点数移动英雄的位置*/

```
/*判断是否单击了人物头像*/
else if(x >0 && x<HERO_FACE_WIDTH && y>HERO_FACE_START_Y && y< HERO_FACE_START_Y+HERO_FACE_HEIGHT){
    if(this.status == 0){                        //是否是待命状态
        this.gvt.setChanging(false);             //暂停变换骰子的线程
        this.diceValue = hero.throwDice(this.diceCount);
        currentSteps = 0;                        //记录要走几格
        for(int i=0;i<diceCount;i++){
            currentSteps = currentSteps+diceValue[i]+1;
        }
        hero.startToGo(currentSteps);
    }
```

/*表示玩家单击的是缩略地图的开关，显示缩略地图的标志位置，然后将缩略地图的 y 坐标设置到屏幕外，使显示缩略地图时到达从屏幕外滑进屏幕的感觉*/

```
}else if(x > MAP_BUTTON_START_X && x< MAP_BUTTON_START_X+MAP_BUTTON_SIZE
        &&y>MAP_BUTTON_START_Y&&y<MAP_BUTTON_START_Y+MAP_BUTTON_SIZE)
{       //如果单击的是地图开关
        showMiniMap = !showMiniMap;
        miniMapStartY = -160;//复位
```

/*当玩家单击第二个骰子或者第三个骰子时，根据当前骰子的个数决定是否显示所单击位置处的骰子*/

```
}else if(x > DICE_START_X+DICE_SPAN && x<DICE_START_X+DICE_SPAN+DICE_SIZE&&
y>DICE_START_Y && y<DICE_START_Y+DICE_SIZE){   //单击第二颗骰子
    if(this.diceCount == 1){                     //当前如果只有 1 个骰子活跃，就变成 2 个
        this.diceCount++;
```

```
}else if(this.diceCount == 2){          //当前如果有 2 个骰子活跃，就变成 1 个
        this.diceCount --;
}
}else if(x > DICE_START_X+DICE_SPAN*2 && x<DICE_START_X+DICE_SPAN*2+DICE_SIZE
        && y>DICE_START_Y && y<DICE_START_Y+DICE_SIZE){//单击第三颗骰子
if(this.diceCount == 2){                //当前如果只有 2 个骰子活跃，就变成 3 个
        this.diceCount ++;
    }else if(this.diceCount == 3){      //当前如果有 3 个骰子活跃，就变成 2 个
        this.diceCount --;
    }}
```

任务 26　游戏界面后台线程 GameViewThread

【任务情境】

GameViewThread 类主要负责定时读取 GameView 的状态，如果为待命，则切换骰子的帧使其产生动画，并且当需要显示缩略地图时，实现缩略地图的滑入效果。

【相关知识】

其代码如下。

```
1    package wyf.ytl;                          //声明包语句
2    public class GameViewThread extends Thread{
3    GameView gv;                              //游戏视图类的引用
4        int sleepSpan = 300;                  //休眠时间
5        int waitSpan = 1500;                  //空转时的等待时间
```

/*以上两行声明两个表示睡眠时间的值：一个是正常的休眠时间，另一个是线程空转的睡眠时间，因为有时不需要转动骰子也不需要绘制缩略地图，只需让该线程空转*/

```
6        boolean flag = true;                  //线程是否执行标志位
7        boolean isChanging;                   //是否需要换骰子动画
8        public GameViewThread(GameView gv){
9    this.gv = gv;
10   }
```

/*线程执行方法*/

```
11       public void run(){
12           while(flag){                      //线程正在执行
13               while(isChanging){            //需要换帧
```

/*需要换骰子时，循环更换骰子的点数。而第 17～22 行为显示缩略地图时，移动缩略地图位置的代码*/

```
14                   int diceNumber = gv.diceCount;
15                   for(int i=0;i<diceNumber;i++){
16                       gv.diceValue[i] = (gv.diceValue[i]+1)%6;          }
```

```
                    if(gv.showMiniMap){              //判断是否显示缩略图
                        gv.miniMapStartY+=30;
                        if(gv.miniMapStartY>=0){    //已经移动到位
                            gv.miniMapStartY = 0;
                        }
                    }
                    try{
                        Thread.sleep(sleepSpan);     //睡眠指定毫秒数
                    }catch(Exception e){//捕获异常
                        e.printStackTrace();         //打印异常信息
                    }
            }try{                                    //不需要换骰子时线程的空转等待时间
                    Thread.sleep(waitSpan);          //睡眠指定毫秒数
                }catch(Exception e){                 //捕获异常
                    e.printStackTrace();             //打印异常信息
                }}}
/*方法：设置是否需要换骰子*/
        public void setChanging(boolean isChanging){
        this.isChanging = isChanging;
    }}
```

【项目小结】

本项目介绍了前台表示层的各个界面，包括ScreenRollView类的开发、ScreenRollThread线程类的开发、游戏界面 GameView、游戏界面绘制方法 onDraw、游戏界面屏幕监听方法onTouch、游戏界面后台线程 GameViewThread，这些实现了英雄角色的相关类功能，对于类似的游戏用处很大，希望读者能熟练掌握。

📖 小知识十五：Android UI 界面构造

UI 界面，对于每个应用而言，它是与用户进行交互的门脸。好的门脸，不只是要亮丽可人，最好还能秀色可餐、过目不忘，甚至还应该有涵养、有气质，彬彬有礼、温柔耐心。

对于开发者来说，锻造这样的面容，不但需要高超的技艺，也需要有顺手的工具。有套好用的 UI 框架，对于开发者而言，是至关重要的。

Android 的 UI 框架，最核心的是资源和 Layout 体系，可通过完善的控件库、简明的接口设计，进一步帮助开发者搭建自己需要的界面。

UI 控件做 UI 就像搭积木，在 Android 中，积木块就是 View。所有其他的 UI 元素，都是派生于此类的子孙类。

图 10-1 可描述 Android 的 UI 控件结构，在每一个窗口下，都是一个标准而完整的树结构。View 有一个子类 ViewGroup，它相当于一个容器类或者是复合控件，所有派生于ViewGroup 的子类，在这棵 UI 树中都可以承担着父节点的职责，而另一些绕过 ViewGroup从 View 直通下来的，就属于叶节点的范畴了。

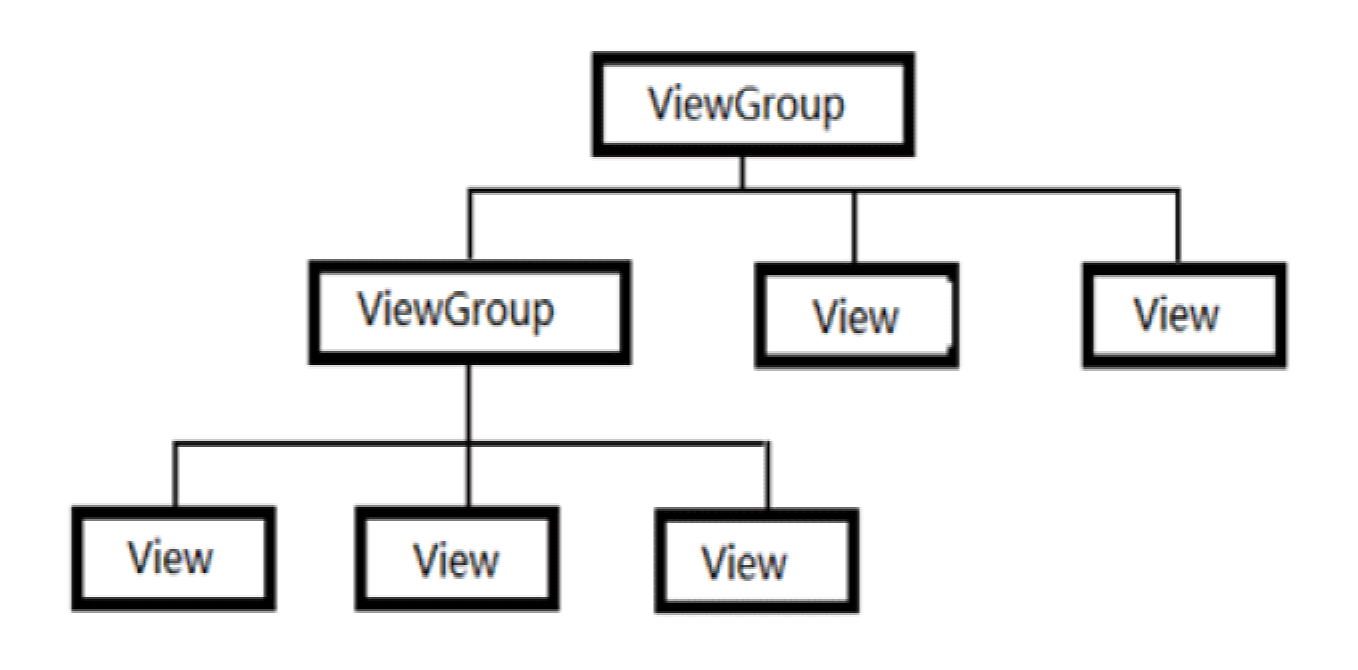


图 10-1　UI 框架

之所以说这是一棵很标准的控件树，是因为父控件对子控件有绝对的掌控权，每个子控件的占地面积和位置，都是基于父控件来分配的，它能够接受和处理的事件，也是父控件派发下去的。这样的结构，被很多平台和框架认可。与传统的 Windows 开发和 Symbian 相比，虽然因为事件传播途径变长了，很多操作的效率变低了，但整个结构更有层次性，每个控件只需要多其父控件负责指挥子控件就好，职责明确、逻辑简单，更有利于开发和设计。

谈及任何平台的控件，都有一些不可避免的主题，比如，每个控件如何标识、如何设定大小和位置、如何接受和处理事件、如何绘制等。

1. 标识

在 Android 中，你可以为每个控件选择设定一个 id，这个 id 的全局的唯一性不需要保证，但在某个局部的范围内具有可识别性，这样就可以通过这个 id 找到这个控件。

但是，在父控件中逐级的 find 比较，找到 id 匹配的控件，然后再做转型，是一个比较重量的操作，于是 Android 又为控件编出另一个属性——tag。它接受任意 object 类型的数据，你可以把与这个控件对象相关的内容堆在里面。比如，在 list 中，我们常常将和每个 list item 相关的控件元素封装成一个 object，扔到 tag 中，就不需要每次都去比较 id 进行寻找，可使工作更加高效、快捷。

2. 尺寸

在 Android 中，控件最重要的大小属性，就是 width/height，开发者可以明确地指明控件的大小，可以设定成为 fill_parent 和 wrap_content，这是概念性的大小。丈量并设定控件的位置，是通过以下两步来进行的。

第一步是 measure。它传入此控件的 width/height 信息，控件会根据自己的参数，计算出真实需要的 width/height，然后调用 setMeasuredDimension 方法，缓存成成员变量，留作后用。

在计算出大小之后，会进行第二步——layout。在这个过程中，父控件会计算其上各个子控件的位置，从而完成整个大小和位置的确定流程。整个 measure 和 layout 的流程，都是自上到下，从树顶往叶子来推进的。

当开发人员需要自定义控件的时候，可能需要关注这些内容，通过重载 onMeasure 和



onLayout 方法，可以定义自己控件的丈量方式。

事件在 Android 中所有的按键、触屏等事件，都是从顶至下进行分发的。每个 ViewGroup 的对象会维系一个 focused 变量，它表示在这个父控件中具备 focus 的控件。当有按键事件发生的时候，会找到这个 focused 子控件，并传递给它。同理，触屏事件的分发也是类似，只不过和 focus 无关，父控件会遍历所有子控件，看看谁处于触碰位置，从而传递给谁。

另外还有一些事件，逻辑上并不是从顶至下发起的。比如，当你修改某个子控件的内容，使得该子控件的大小和内容都发生了变化，就需要进行控件的重排和重绘，这些操作不仅是子控件自己的事情，甚至需要整个控件树上的所有控件来配合。在 Android 中，处理这类事情的实现策略是子控件维系一个 ViewParent 对象，该对象象征着整个控件树的管理者，子控件产生影响整个控件树的事件时，会通知到 ViewParent，ViewParent 会将其转换成一个自顶向下的事件，分发下去。

Android 的事件处理逻辑，采用的是观察者模式。Android 的控件提供了一些列的 add/set Listener 的接口，使得外部观察者有机会处理控件事件。比如，你需要在某个 button 被单击时做一些事情，你就需要派生一个 View.OnClickListener 对象作为观察者，调用该控件的 setOnClickListener 接口注册进去，当 button 被单击，就可以获得处理单击事件的机会了。当然，有的时候，你需要处理的逻辑更为复杂，光是站在外面围观叫好不能解决问题，可能还需要派生某个控件，去重载 onXXXX 之类的事件处理函数，来进行更完整的控制。

3. 焦点

对于一个非触屏的机器，焦点的维系是一个极其重要的事情，而在有触屏的年代，焦点的地位虽有所下降，但依然还是需要妥善保护的。

Android 中，是以控件树为单位来管理焦点的。每个控件，可以设置上下左右四向的 focus 转移对象。当在一个控件上发生焦点转移事件时，Android 会如前述，自顶向下根据设定好的焦点转移逻辑，跳转到正确的控件上。

4. Layout

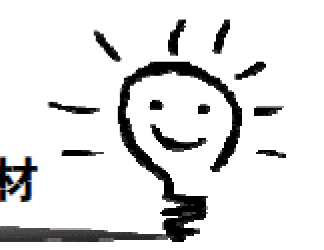
Layout 是一类特殊的 ViewGroup 控件，它们本身没有任何可显示内容，如同透明的玻璃盒子，存活的唯一理由，就是其的内部结构，能够更好的摆放它的子控件们。

比如线性的 Layout、LinearLayout。放入这个 Layout 的子控件，会按水平或垂直方向，一个挨着一个按顺序排列。TableLayout 可以将子控件按照表格的形式，一枚枚放置好。而 RelativeLayout 则更灵活，可以设定各个控件之间的对齐和排列关系，适合定制复杂的界面。

有了 Layout 的存在，控件和控件之间不再割裂地存在，而是更有机地结合在了一起，设定起来也更为方便。比 Symbian 维系各个控件的关系，轻松自在多了。

5. 更多

这些问题的完整答案，可参见 SDK 中 View 的页面：/reference/android/view/View.html。



综合实训十 微博随身之个人中心模块的实现

【问题情境】

在用户登录或注册成功后，可以进入个人中心。个人中心界面主要由 FunctionTabActivity 实现，下面将介绍该 Activity 的开发。

【拓展知识】

1. 个人中心界面的开发

下面将介绍个人中心界面的开发，FunctionTabActivity 继承自 TabActivity，主要的功能是向用户列出程序的各个功能，其代码如下。

```
package wyf.wpf;                                           //声明包语句
import android.app.AlertDialog;                            //引入相关类
...//此处省略部分引入相关类的代码
import android.widget.TabHost;                             //引入相关类
public class FunctionTabActivity extends TabActivity{
    static final int MENU_SEARCH = 0;                      //声明菜单项编号
    static final int MENU_EXIT = 1;
    String uno = null;                                     //声明用于存放用户 ID 的成员变量
    protected void onCreate(Bundle savedInstanceState) {   //重写 onCreate 方法
        super.onCreate(savedInstanceState);
        Intent intent = getIntent();                       //获得启动该 Activity 的 Intent
        uno = intent.getStringExtra("uno");                //读取“uno”字段
        final TabHost tabHost = getTabHost();              //获得 TabHost
        Intent intentPublish = new Intent(this,wyf.wpf.PublishActivity.class);
        intentPublish.putExtra("uno", uno);                //为将用户 ID 设置为 Intent 的 Extra
        Intent intent1 = new Intent(this,wyf.wpf.ContactsActivity.class);
        intent1.putExtra("type", 0);intent1.putExtra("uno", uno);  //type 为 1 表示好友列表
        Intent intent2 = new Intent(this,wyf.wpf.ContactsActivity.class);
        intent2.putExtra("type", 1);intent2.putExtra("uno", uno);  //type 为 2 表示最近访客
        Intent iDiary = new Intent(this,wyf.wpf.MyDiaryActivity.class);
        iDiary.putExtra("uno", uno);                       //为将用户 id 设置为 Intent 的 Extra
        Intent iAlbum = new Intent(this,wyf.wpf.MyAlbumListActivity.class);
        iAlbum.putExtra("uno", uno);
        tabHost.addTab(tabHost.newTabSpec("tab1")
                .setIndicator("快速发布", getResources().getDrawable(R.drawable.publish))
                .setContent(intentPublish));               //添加“快速发表”选项卡
        tabHost.addTab(tabHost.newTabSpec("tab2")"
                .setIndicator("我的好友", getResources().getDrawable(R.drawable.friend))
                .setContent(intent1));                     //添加“我的好友”选项卡
```

```
30            tabHost.addTab(tabHost.newTabSpec("tab3")
31                    .setIndicator("最近访客", getResources().getDrawable(R.drawable.visitor))
32                    .setContent(intent2));              //添加“最近访客”选项卡
33            tabHost.addTab(tabHost.newTabSpec("tab4")
34                    .setIndicator("日志列表",getResources().getDrawable(R.drawable.diary))
35                    .setContent(iDiary));               //添加“日志列表”选项卡
36            tabHost.addTab(tabHost.newTabSpec("tab5")
37                    .setIndicator("相册列表", getResources().getDrawable(R.drawable.album))
38                    .setContent(iAlbum));               //添加“相册列表”选项卡
39            String tab = intent.getStringExtra("tab");
40            if(tab != null){tabHost.setCurrentTabByTag(tab);
41        }
42        public boolean onCreateOptionsMenu(Menu menu) {     //显示菜单的回调方法
43        public boolean onOptionsItemSelected(MenuItem item) {
          //处理菜单选项被选中时间的回调方法
44    }
```

☑ 第 14～23 行代码声明了一系列的 Intent 对象，每个 Intent 对象都将用来启动相应的 Activity。这些 Intent 对象将会作为 TabActivity 中选项卡的显示内容，即当单击某个选项卡时，会通知相应的 Intent 对象启动 Activity 并将其显示到该选项卡中。

☑ 第 24～38 行为 TabActivity 添加了 tab1～tab5 共 5 个选项卡，并为这些选项卡设置了图标和说明文字，以及按下后启动的 Activity。

☑ 第 39～40 行为读取启动该 FunctionTabActivity 的 Intent 中的“tab”字段的值，这段代码的功能是设置启动 FunctionTabActivity 后首先显示的选项卡。例如，用户从日志边界界面返回到个人中心界面时，通过在 Intent 中设置“tab”的值可以让个人中心界面直接显示日志列表，这样比较符合操作习惯。

2. 个人中心界面菜单功能的开发

个人中心除了将主要功能以选项卡的形式显示之外，还为其提供了两个菜单选项：搜索和退出，这两项菜单选项通过重新 onCreateOptionsMenu 方法来创建，其代码如下。

```
public boolean onCreateOptionsMenu(Menu menu) {
menu.add(0, MENU_SEARCH, 0, "搜索").setIcon(R.drawable.search);
menu.add(0, MENU_EXIT, 0, "退出").setIcon(R.drawable.exit);
return super.onCreateOptionsMenu(menu);
}
```

说明

上述代码添加了两个菜单选项，当按下手机键盘上的“MENU”键时，会弹出这两个菜单选项，onCreatOptionMenu 仅在第一次显示菜单时被调用。

添加好菜单选项，还需要开发菜单选项被选中事件的处理代码，这些代码需要写在 onOptionItemSelected 方法中。

```
1    public boolean onOptionsItemSelected(MenuItem item) {
2            switch(item.getItemId()){                        //判断单击的菜单选项是哪个
3            case MENU_SEARCH:                                //单击“搜索”菜单选项
4                Intent intent = new Intent(this,SearchActivity.class);      //创建 Intent
5                intent.putExtra("visitor", uno);              //设置 Intent 的 Extra 字段
6                startActivity(intent);                       //启动 Activity
7                break;
8            case MENU_EXIT:                                  //单击“退出”菜单选项
9                new AlertDialog.Builder(this).setTitle("提示")
10               .setMessage("确认退出吗？")                  //设置对话框显示信息
11               .setIcon(R.drawable.alert_icon)              //设置对话框显示内容
12               .setPositiveButton("确定", new DialogInterface.OnClickListener() {
13                   //为按钮添加监听器
14                   public void onClick(DialogInterface dialog, int which) {
15                   android.os.Process.killProcess(android.os.Process.myPid());//结束进程
16               }})
17               .setNegativeButton("取消", new DialogInterface.OnClickListener() {
18                   //添加“取消”按钮
19                   public void onClick(DialogInterface dialog, int which) {}
20                   }).show();                               //显示对话框
21               break;
22           }
23           return super.onOptionsItemSelected(item);
24       }
```

☑ 第 3～7 行为选中“搜索”菜单选项后执行的代码，首先创建一个指向 SearchActivity 的 Intent 对象，并为其设置 Extra 属性值，然后通过 startActivity 方法启动 SearchActivity。

☑ 第 9～21 行为选中“退出”菜单选项后执行的代码，按下“退出”菜单后将会显示一个 AlertDialog，该对话框中提示用户是否确认退出，用户可以单击“确定”或“取消”按钮来选择是否退出应用程序。

小知识十六：MSNS 开发概述

与传统的 SNS 应用相比，MSNS 应用最明显的差别是：客户端开发平台并不是浏览器或 PC，而是各种各样的手机，因此在客户端开发上 MSNS 应用有很大不同。

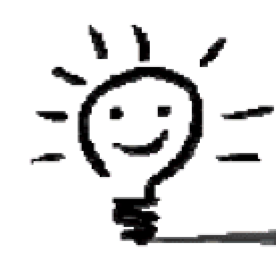
MSNS 应用开发一般具有以下 7 个步骤：

（1）确定功能需求

在进行开发之前，应首先设计好该应用包括哪些模块，每个模块中包括哪些子功能，然后才能依次进行设计、开发。

（2）设计功能接口

MSNS 应用一般为客户端-服务器结构（Client-Server model），因而，针对每个子功能设计一个完善的接口是必需的，这样才能做到服务器端与客户端开发的分离。



（3）设计页面逻辑

在功能需求确定之后，就可以着手设计页面跳转逻辑了，大致步骤为：

① 整个应用需要哪些页面。

② 每个页面包含哪些功能。

③ 每个页面可以执行哪些操作。

④ 执行操作后，页面应该如何直接跳转。

在进行逻辑设计时应做到：

① 页面层次清晰。

② 页面跳转明了。

③ 页面功能明了。

（4）设计界面 UI

UI（User Interface，用户界面）。界面设计则是指对软件的人机交互、操作逻辑、界面美观的整体设计。好的 UI 设计不仅可以使软件变得更有个性、品位，还要使软件的操作变得更舒适、简单、自由，能充分体现软件的定位和特点。

手机的易携带性给用户带来了方便的同时，其小尺寸的屏幕也让用户的操作范围大大减少了，因此在 UI 设计上与 PC 的有所不同。另外，由于当前的智能手机一般均采取触摸屏，在设计中也应该加以注意。触摸屏有横屏和竖屏两种模式，所开发的应用是否涉及到横、竖屏模式的切换，以及相应模式的不同设计方法，都是 UI 设计者需要考虑的问题。

（5）设计数据库

数据库设计是指对于一个给定的应用环境，构造最优的数据库模式，建立数据库及其应用系统，使之能够有效地存储数据，满足各种用户的应用需求（信息要求和处理要求）。

数据库设计是 C-S 应用的基础，一个完善的数据库设计能够使之后的应用开发事半功倍。

（6）服务器端程序开发

在上述工作完成之后，服务器端程序和手机端程序就可以进行开发了。应根据一般的服务器开发原则，开发满足接口要求的服务器程序。

（7）手机端程序开发

在进行手机端程序开发时，应尽量做到服务器交互、界面实现的分离，做好模块化开发。并注意以下几点：

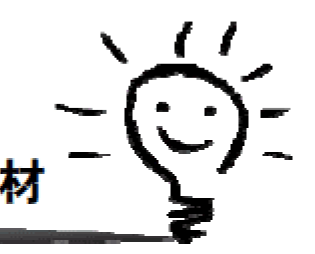
① 处理器频率低，也就意味着只能执行小量的运算，因此算法要尽量精练、简洁。

② 上网速度慢、上网资费高，尽管已经步入 3G 时代，但不同时段、不同地点也会出现上网慢的现象，因此要做到尽量少的交互和缓存功能。

③ 输入麻烦，选择也麻烦，因此在开发设计时要对用户的输入方式、页面跳转进行事先考虑。

④ 合理利用系统提供的标准控件。

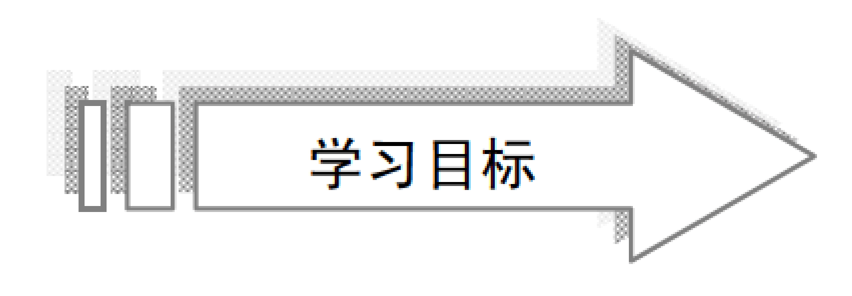
当前的智能手机操作系统，基本都提供了封装好的标准功能接口，开发时要尽量调用这些接口，而不要重新开发。



项目十一
管理面板模块的开发



前面对游戏的主界面以及其相关界面进行了介绍，本项目将对游戏过程中的各个管理面板界面进行简单介绍，这些管理面板的作用是查询或管理游戏的相关信息，如城池、粮草、将领信息等。



- 掌握人物属性面板类 ManPaneView 的开发方法
- 掌握城池管理面板类 CityManageView 的开发方法

任务 27 人物属性面板类 ManPaneView 的开发

【任务情境】

首先介绍的是人物属性界面，该界面为玩家单击主界面中的“人”按钮后弹出的界面，主要作用是显示英雄的基本属性及其所拥有的技能。

【相关知识】

任务属性面板的运行效果在游戏开始时做过介绍。这里对该界面的开发步骤做出简单介绍，步骤如下。

（1）框架的搭建，其代码如下。

```
package wyf.ytl;
import java.util.HashMap;                          //引入 HashMap 类
import java.util.Set;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.view.MotionEvent;                   //引入 MotionEvent 类
public class ManPanelView{
    GameView gameView;                             //gameView 的引用
    int yeSpan = 11;                               //每页显示的个数
    int yeSpan2 = 10;                              //每页显示的个数
/*以上两行的两个变量分别表示基本属性界面以及技能属性界面每页最多显示的数量*/
    int currentI = 0;                              //当前绘制的屏幕最上边的元素下标
    int selectI = 0;                               //当前选中行数
/*以上两行的两个变量则表示当前屏幕中最上面信息的下标以及当前选中信息的下标*/
    int startY = 120;                              //y 坐标
    int startX = 25;
    int status = 0;                                //显示的标记位，0-属性列表   1-技能列表
    String[][] items1;                             //需要列出的属性
    String[][] items2;                             //需要列出的技能
    static Bitmap threeBitmap;                     //右上角的 3 个按钮
    static Bitmap panel_back;                      //背景图
    static Bitmap selectBackground;                //选中后的背景
    static Bitmap buttonBackGround;                //按钮背景
    static Bitmap upBitmap;                        //向上的小箭头
    static Bitmap downBitmap;                      //向下的小箭头
    static Bitmap menuTitle;                       //表格的标题背景
```

```
        static Bitmap logo;
        Paint paint;
    /*该类的构造方法，在方法中对画笔进行初始化并设置画笔颜色等，同时调用数据初始化方法对数据
进行初始化*/
        public ManPanelView(GameView gameView) {
            this.gameView = gameView;
            paint = new Paint();
            paint.setARGB(255, 42, 48, 103);                //设置字体颜色
            paint.setAntiAlias(true);                       //抗锯齿
            initData();                                     //初始化数据
        }
    /*图片资源初始化的方法，在该方法中对该界面所用到的图片进行初始化*/
    public static void initBitmap(Resources r){             //初始化图片资源
        ...//该处省略了图片资源的初始化工作
    }
    /*对界面需要的数据进行初始化*/
    public void initData(){                                 //初始化数据
        ...//该处省略了该界面所用到的数据的初始化工作，将在后面进行介绍
    }
    /*该界面的绘制方法，在介绍 GameView 的 onDraw 方法时，已经看到当 GameView 中状态值表示当
前显示的是人物属性界面，直接调用人物属性界面的 onDraw 方法进行绘制*/
    public void onDraw(Canvas canvas){                      //绘制的方法
        ...//该处省略了界面绘制的代码，将在后面进行介绍
    }
    /*屏幕监听方法，同样是在 GameView 的屏幕监听方法中被调用*/
    public boolean onTouchEvent(MotionEvent event) {
        ...//该处省略了该界面的屏幕监听方法，将在后面进行介绍
    }}
```

（2）初始化数据方法 initData 的实现，其代码如下。

```
    public void initData(){                                 //初始化数据的方法
    /*根据 gameView 中的 hero 对象初始化人物属性界面需要的相关信息，并将这些信息整理到字符串
数据中*/
        this.items1 = new String[][]                        //初始化人物属性
        {
            {"名称:", gameView.hero.getName()},              //英雄的名称
            {"等级:", gameView.hero.getLevel()+" 级"},        //英雄的等级
            {"官衔:", gameView.hero.getTitle()},             //英雄的官衔
            {"将军:", gameView.hero.getGeneralNumber()+" 个"}, //将军的个数
            {"城池:", gameView.hero.getCityList().size()+" 个"}, //城池的个数
            {"黄金:", gameView.hero.getTotalMoney()+" 金"},   //黄金的数量
            {"粮草:", gameView.hero.getTotalFood()+" 石"},    //粮草的数量
            {"兵力:", gameView.hero.getTotalArmy()+" "},      //兵力的数量
            {"总人口:", gameView.hero.getTotalCitizen()+" 人"}, //总人口的数量
            {"投石车:", gameView.hero.getTotalWarTank()+" 个"}, //投石车的个数
```

```
            {"箭垛:", gameView.hero.getTotalWarTower()+" 个"}, //箭垛的数量
        };
/*对英雄拥有的技能进行循环，得到每个技能，并将其信息存放到数组中*/
        /*初始化技能*/
        HashMap<Integer,Skill> heroSkill = gameView.hero.getHeroSkill(); //得到英雄的所有技能
        items2 = new String[heroSkill.size()][3];                        //初始化 item2 数组
        Set<Integer> s = heroSkill.keySet();                             //得到 HashMap 键的集合
        int k = 0;
        for(Integer i : s){                                              //循环
            Skill skill = heroSkill.get(i);                              //得到技能
            items2[k][0] = skill.getName();                              //技能名字
            items2[k][1] = skill.getProficiencyLevel()+"";               //技能等级
            items2[k][2] = skill.getStrengthCost()+"";                   //技能消耗的体力
            k++;
        }}
```

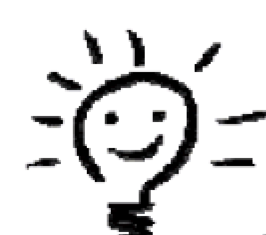
（3）屏幕绘制方法 onDraw 的实现，其代码如下。

```
public void onDraw(Canvas canvas){                                      //绘制的方法
/*绘制人物属性界面公共的部分*/
        canvas.drawBitmap(panel_back, 0, 0, paint);                     //在指定位置绘制图片
        canvas.drawBitmap(threeBitmap, 212, 15, paint);
        canvas.drawBitmap(logo, 15, 16, paint);                         //绘制 logo
        paint.setTextSize(23);                                          //设置文字大小
        canvas.drawText("个人属性", 50, 40, paint);
        paint.setTextSize(18);                                          //设置文字大小
        canvas.drawBitmap(buttonBackGround, 15, 57, paint);
        canvas.drawBitmap(buttonBackGround, 85, 57, paint);
        canvas.drawText("属性", 27, 78, paint);
        canvas.drawText("技能", 97, 78, paint);                          //在指定位置绘制文字
/*绘制人物属性界面的代码，各部分具体作用见以下注释*/
        if(status == 0){                                                //画属性界面时
            int tempCurrentI = currentI;
            if(items1.length < yeSpan){                                 //当一个屏幕可以显示全时
                for(int i=0; i<items1.length; i++){
                    for(int j=0; j<items1[i].length; j++){
                        canvas.drawText(items1[i][j], startX+j*115, startY + 30*i, paint);
                    }
                }
/*以上为一个屏幕可以显示下所有信息的情况，即不需要分页显示*/
            }else{                                                      //当一个屏幕显示不下时
                for(int i=tempCurrentI; i<tempCurrentI+yeSpan; i++){
                    for(int j=0; j<items1[i].length; j++){
                        canvas.drawText(items1[i][j], startX+j*115, startY + 30*(i-tempCurrentI), paint);
                }}}
/*以上为需要分页的情况*/
```

```
canvas.drawBitmap(selectBackground, 10, startY + 30*selectI - 22, paint);
/*选中后的效果*/
        if(tempCurrentI != 0){
            canvas.drawBitmap(upBitmap, 150, 85, paint);          //绘制小的向上箭头
        }
        if(items1.length>yeSpan && (tempCurrentI+yeSpan) < items1.length){
            canvas.drawBitmap(downBitmap, 150, 455, paint);       //绘制小的向下箭头
        }
/*以上根据一页是否显示完全来绘制上下小箭头，来表示上面或者下面是否还有信息*/
/*绘制技能信息的代码，首先绘制界面头，之后的绘制与人物基本属性的界面基本相同，根据是否分
页采用不同的绘制方法*/
    }else if(status == 1){                                        //绘制技能界面
        canvas.drawBitmap(menuTitle, 10, 100, paint);             //绘制菜单头
        canvas.drawText("技能名", 15, 120, paint);                //绘制文字“技能名”
        canvas.drawText("等级", 125, 120, paint);                 //绘制文字“等级”
        canvas.drawText("消耗", 240, 120, paint);                 //绘制文字“消耗”
        int tempCurrentI = currentI;
        if(items2.length < yeSpan2){                              //当一个屏幕可以显示全时
            for(int i=0; i<items2.length; i++){
            canvas.drawText(items2[i][0], 15, startY + 35 + 30*i, paint);   //绘制文字
            canvas.drawText(items2[i][1], 125, startY + 35 + 30*i, paint);  //绘制文字
            canvas.drawText(items2[i][2], 240, startY + 35 + 30*i, paint);  //绘制文字
            }
        }else{                                                    //当一个屏幕显示不全时
            for(int i=tempCurrentI; i<tempCurrentI+yeSpan2; i++){
                canvas.drawText(items2[i][0], 15, startY + 35 + 30*(i-tempCurrentI), paint);
                canvas.drawText(items2[i][1], 125, startY + 35 + 30*(i-tempCurrentI), paint);
                canvas.drawText(items2[i][2], 240, startY + 35 + 30*(i-tempCurrentI), paint);
                }
            }
            canvas.drawBitmap(selectBackground, 10, startY + 35 + 30*selectI - 22, paint);
            if(tempCurrentI != 0){
                canvas.drawBitmap(upBitmap, 150, 85, paint);      //绘制图片
            }
            if(items2.length>yeSpan2 && (tempCurrentI+yeSpan2) < items2.length){
                canvas.drawBitmap(downBitmap, 150, 455, paint);   //绘制图片
            }}
*/
```

（4）屏幕监听方法 onTouchEvent 的实现，其代码如下。

```
public boolean onTouchEvent(MotionEvent event) {
    if(event.getAction() == MotionEvent.ACTION_DOWN){             //屏幕被按下
        int x = (int) event.getX();                               //得到 x 坐标
        int y = (int) event.getY();                               //得到 y 坐标
/*单击向下翻页按钮的处理代码，其中根据一屏是否能够显示全进行分类处理*/
```



```
if(y>15 && y<45){
    if(x>212 && x<242){                              //单击了向下翻页按钮
        if(status == 0){                             //人物基本属性时
            selectI++;                               //选中下一条信息
            if(items1.length < yeSpan){ //当一个屏幕可以全部显示时，即不需要滚屏
                if(selectI > items1.length-1){
                    selectI = items1.length-1;       //当到页面最下面时
                }
            }else {                  //当一屏显示不全，需要滚屏时
                if(selectI > yeSpan-1){              //当到页面最下面时
                    selectI = yeSpan-1;              //选中下一个
                    currentI++;                      //滚动屏幕
                    if((currentI+yeSpan) > items1.length){
                        currentI--;                  //恢复
                    }}}
        }else if(status == 1){                       //技能显示时
            selectI++;                               //显示下一条
            if(items2.length < yeSpan2){//当一个屏幕可以全部显示时，即不需要滚屏
            if(selectI > items2.length-1){           //到达最后一条信息时
                selectI = items2.length-1;
            }
            }else {                  //当一屏显示不全，需要滚屏时
                if(selectI > yeSpan2-1){             //到达页面最下面时
                    selectI = yeSpan2-1;             //选中下一个
                    currentI++;                      //滚动屏幕
                    if((currentI+yeSpan2) > items2.length){
                        currentI--;                  //恢复
                    }}}}
/*单击的是向上翻页按钮，选中上一条，而当到达第一条时不移动*/
    }else if(x>243 && x<273){                        //单击了向上翻页按钮
        selectI--;
        if(selectI < 0){                             //单击了向上翻页按钮
            selectI = 0;                             //当到达最前面时
            currentI--;
            if(currentI < 0){                        //选中第一条
                currentI = 0;                        //设置为第一条
            }}
/*表示玩家单击的是“关闭”按钮，将所有变量恢复到 0 状态，然后设置 gameView 的状态值到游戏状态*/
/*“属性”按钮的事件监听，将状态置成 0 并选中第一条信息，同时将屏幕滚动到最上面*/
    }else if(x>274 && x<304){                        //单击了“关闭”按钮
        this.status = 0;                             //恢复状态值到 0
        this.selectI = 0;                            //选中第一个
        this.currentI = 0;                           //滚屏到第一条信息
        gameView.setStatus(0);                       //设置 gameView 的状态值
    }
```

```
/*“技能”按钮的事件监听，将状态设置成 1 并选中第一条信息，同时也将屏幕滚动到最上面*/
            }else if(y>57 && y<57+buttonBackGround.getHeight() && x>15 && x<15 + buttonBack
Ground.getWidth()){                                              //单击了“属性”按钮
                this.status = 0;//恢复状态值到 0
                this.selectI = 0;                                //选中第一个
                this.currentI = 0;                               //滚屏到第一条信息
            }else if(y>57 && y<57+buttonBackGround.getHeight() && x>85 && x<85 + buttonBack
Ground.getWidth()){                                              //单击了“技能”按钮
                this.status = 1;                                 //设置状态值到 1
                this.selectI = 0;                                //选中第一个
                this.currentI = 0;                               //滚屏到第一条信息
            }}
        return true;
    }
```

提示

其中 selectI 与 currentI 分别表示当前屏幕中所选中的信息编号与当前屏幕最上方的信息在所有信息中的编号。

任务 28 城池管理面板类 CityManageView 的开发

【任务情境】

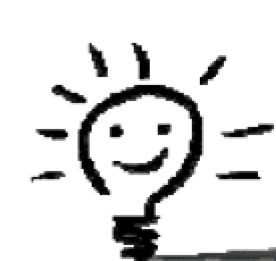
前面已经介绍了人物属性界面的开发，下面将介绍另一个控制面板——城池管理界面的开发，该界面主要是对英雄所拥有的城池进行管理，例如，粮草的分配、兵力的划拨等。

【相关知识】

接下来对该界面的开发步骤做出简单介绍，步骤如下。

（1）框架的搭建，其代码如下。

```
package wyf.ytl;
import java.util.ArrayList;                          //引入相关类
import android.content.res.Resources;                //引入相关类
import android.graphics.Bitmap;                      //引入相关类
import android.graphics.BitmapFactory;               //引入相关类
import android.graphics.Canvas;                      //引入相关类
import android.graphics.Paint;                       //引入相关类
import android.view.MotionEvent;                     //引入相关类
public class CityManageView {
    GameView gameView;
```



```
        int yeSpan1 = 9;                              //主页显示城池的个数
        int yeSpan2 = 9;                              //管理页显示城池的个数
        int currentI = 0;                             //当前绘制的屏幕最上边的元素下标
        int selectI = 0;
        int startY = 120;
        int startX = 17;
        int status = 0;                               //0-主界面   1-详细信息
        String[][] items1;                            //城池列表
        String[][] items2;                            //城池管理界面列表
        ArrayList<CityDrawable> cityList;             //存放英雄拥有的城池
    /*ArrayList 存储的是英雄所拥有的所有城池*/
        CityDrawable selectCityDrawable;              //当前选中的城市
        static Bitmap menuTitle;                      //表格的标题背景
        static Bitmap threeBitmap;                    //右上角的三个按钮
        static Bitmap panel_back;                     //背景图
        static Bitmap selectBackground;               //选中后的背景
        static Bitmap buttonBackGround;               //按钮背景
        static Bitmap upBitmap;                       //向上的小箭头
        static Bitmap downBitmap;                     //向下的小箭头
        static Bitmap addBitmap;                      //小加号
        static Bitmap cutBitmap;                      //小减号
        static Bitmap logo;
        Paint paint;                                  //画笔
        public CityManageView(GameView gameView) {    //构造器
            ...//该处省略了构造方法中的相关代码
        }
    /*上述构造方法中先对画笔进行初始化并做相应设置，然后调用初始化数据方法 initData 初始化相关
数据*/
        public static void initBitmap(Resources r){   //初始化所有图片资源
            ...//该处省略了图片资源的初始化代码
        }
        public void initData(){//该界面中数据的初始化方法
            ...//该处省略了数据的初始化工具
        }
        public void onDraw(Canvas canvas){            //绘制的方法
            ...//该处省略了屏幕的绘制工作
    }
    /*以上为界面的绘制方法，在方法中会根据该类中当前的状态值绘制不同的界面*/
    public boolean onTouchEvent(MotionEvent event){
        ...//该处省略了屏幕事件的处理代码
    }
    //以上为界面的屏幕监听方法，根据当前的状态值以及玩家单击的位置做出相应的处理
    public void initItems2(){
        ...//该处省略了城池详细信息的初始化工具
    }
    /*以上代码根据玩家选择的城池初始化城池的详细信息*/
```

（2）接下来介绍城池详细信息的初始化方法 initItem2，其代码如下。

```
public void initItems2(){                                              //初始化选中城池的信息
    selectCityDrawable = cityList.get((selectI+currentI));             //得到选中的城池
    items2 = new String[][]{
        {"名称：", selectCityDrawable.getCityName(),"",""},              //城池的名称
        {"等级：", selectCityDrawable.getLevel()+"","",""},              //城池的等级
        {"兵力：", selectCityDrawable.getArmy()+"","剩余：",gameView.hero.getArmyWithMe()+""},
        {"粮草：", selectCityDrawable.getFood()+"", "剩余：",gameView.hero.getFood()+""},
        {"防御：", GameFormula.getCityDefence(selectCityDrawable)+"","",""},
        {"攻击：", GameFormula.getCityAttack(selectCityDrawable)+"","",""},
        {"居民：", selectCityDrawable.getCitizen()+"","","","",""},      //城池的居民人口
        {"箭垛：", selectCityDrawable.getWarTower()+"","剩余：",gameView.hero.warTower+""},
        {"战车：", selectCityDrawable.getWarTank()+"","剩余：",gameView.hero.warTank+""     }
    };}
```

（3）屏幕监听方法 onTouchEvent 的实现，在该方法中，根据状态值的不同分为两部分：第一部分为城池列表界面的处理代码，另一部分为某个城池管理界面的处理代码。城池列表界面的处理方式与人物属性界面的基本相同。下面给出的是城池管理界面的处理代码。

```
else if(status == 1){                                                  //城池管理界面时
/*加兵力按钮的事件处理代码，先判断英雄的兵力是否大于 500，当大于 500 时划拨 500 兵力到选中
的城池，而不足 500 时将全部兵力划拨到选中的城池中*/
    if(x>250 && x<265 && y>130 && y<145){                              //加兵力
        if(gameView.hero.getArmyWithMe()>=500){                        //身上的并大于 500 时
            gameView.hero.setArmyWithMe(gameView.hero.getArmyWithMe()-500);
            this.selectCityDrawable.setArmy(this.selectCityDrawable.getArmy()+500);
        }else {
            this.selectCityDrawable.setArmy(this.selectCityDrawable.getArmy()+gameView.hero.getArmy
            WithMe());gameView.hero.setArmyWithMe(0);                  //置空
        }
initItems2();                                                          //初始化城池的详细信息
/*减兵力按钮的事件处理代码，此时需要先判断选择的城池中的兵力是否大于 500，因为需要从城池
中划拨 500 兵力跟随英雄，而不足 500 时，需要将全部兵力划拨到英雄身边*/
}else if(x>270 && x<285 && y>130 && y<145){                            //减兵力
    if(this.selectCityDrawable.getArmy()>=500){                        //当大于 500 时
    gameView.hero.setArmyWithMe(gameView.hero.getArmyWithMe()+500);
    this.selectCityDrawable.setArmy(this.selectCityDrawable.getArmy()-500);
}else {
    gameView.hero.setArmyWithMe(gameView.hero.getArmyWithMe()+this.selectCityDrawable.getAr
my());                                                                 //当不足 500 时
this.selectCityDrawable.setArmy(0);                                    //设置为 0
}
initItems2();                                                          //初始化城池的详细信息
...//该处省略了处理箭垛、粮草划拨的代码
/*处理战车调拨的代码，其原理与划拨兵力的相同，在此就不再赘述*/
}else if(x>250 && x<265 && y>310 && y<325){                            //加战车
```

```
        if(gameView.hero.getWarTank()>0){                    //当只有当英雄身上有战车时
        this.selectCityDrawable.setWarTank(this.selectCityDrawable.getWarTank()+1);
        gameView.hero.setWarTank(gameView.hero.getWarTank()-1);
    }
        initItems2();                                        //初始化城池的详细信息
    }else if(x>270 && x<285 && y>310 && y<325){              //减战车
        if(this.selectCityDrawable.getWarTank()>0){          //当城池中有战车时
        gameView.hero.setWarTank(gameView.hero.getWarTank()+1);
        this.selectCityDrawable.setWarTank(this.selectCityDrawable.getWarTank()-1);
        }
        initItems2();                                        //初始化城池的详细信息
    }}
```

提示

其他管理面板的实现技术与前面介绍的两个面板类基本相同，因篇幅有限，在此就不再介绍。

【项目小结】

本项目介绍了游戏过程的各个管理面板界面，其作用是查询或管理游戏的相关信息，主要知识包括人物属性面板类 ManPanelView 的开发、城池管理面板类 CityManageView 的开发方法，读者需要充分理解并熟练掌握本项目内容，可以用于相关游戏的类似管理面板的开发过程。

小知识十七：Android 游戏开发起步

Android 是一个基于 Java 的环境。这对初学者来说是个好消息，因为相对于 C++，Java 被广泛认为是一门更容易上手的语言，它是移动开发的规范。此外，Google 也做了一件出色的工作，它将 API 文档化并提供示例代码供使用。其中有个叫做 API Demos 的示例几乎展示了所有 API 的功能。如果你熟悉 Java 并且用过 Eclipse，要让你第一个应用起来相当简单。如果你以前从没写过代码，在你前进路上还要学习很多，但别气馁。

1. 获取 SDK

新手上路的第一步便是获取 Android SDK（软件开发工具包）。SDK 里有一个核心类库、一个模拟器、一些工具和示例代码。我强烈建议使用 Eclipse 和 Android Eclipse 插件。如果你玩 Android 的话，Eclipse IDE 对 Java 开发者来说很好用。如果这是你第一次开发 Java 项目，你可能会需要下载全套 JDK，它里面包括签名和部署你的应用程序的一些工具。

别急着一头扎进开发的海洋里，理解 Android 应用程序架构是很重要的。如果你不学一下，你设计出来的游戏在线下将很难调试。你将需要理解 Applications、Activities、Intents 以及它们怎样相互联系。Google 提供了很多有用的架构信息。真正重要的是要理解为什么你的游戏需要多于一个的 Activity，以及什么才是设计一个有良好用户体验的游戏。要理解这些，首先要了解什么是 Activity 生命周期。

2. 学习 Activity 生命周期

Activity 生命周期由 Android 操作系统来管理。你的 Activity 的创建、恢复、暂停、销毁都受操作系统的支配。正确处理这些事件是很重要的，这样应用程序才能表现良好，做用户认为正确的事。在设计游戏之前了解这些是如何工作的可以节省调试时间和昂贵的重新设计时间。对大多数应用来说，默认的设置将工作正常，但对于游戏，你可能需要考虑将 SingleInstance 标志打开。当设置为默认时，Android 在它认为合适时会创建 activity 的新实例。对于游戏来说，你可能只需要一个游戏 Activity 的实例。这对于你要怎样管理事务的状态有些影响，但对于我来说，这解决了一些资源管理的问题，应予以考虑。

3. 主循环

根据要写的游戏的类型，你可能需要/不需要一个主循环。如果你的游戏不依赖于时间或者它仅仅对用户所做的加以回应，并且不做任何视觉上的改变，永远等待着用户的输入，那么你就不需要主循环。如果你写的是动作类游戏或者带有动画、定时器或任何自动操作的游戏，你应该认真考虑下使用主循环。

游戏的主循环以一个特定的顺序通常尽可能多地在每秒钟内“滴答”提醒子系统运行。你的主循环需要在它自己的线程里运行，原因是 Android 有一个主用户界面线程，如果你不运行自己的线程，用户界面线程将会被你的游戏所阻塞，这会导致 Android 操作系统无法正常的更新任务。执行的顺序通常如下：状态→输入→人工智能→物理→动画→声音→录像。

更新状态的意思是管理状态转换，例如游戏的结束、人物的选择或下一个级别。很多时候你需要在某个状态上等待几秒钟，而状态管理应该处理这种延迟，并且在时间过了之后设置成下一个状态。

输入是指用户按下的任何键、对于滚动条的移动或者触摸。在处理物理之前处理这些是很重要的，因为很多时候输入会影响到物理层，因而首先处理输入将会使游戏的反应更加良好。在 Android 里，输入事件从主用户界面线程而来，因此你必须写代码将输入放入缓冲区，这样你的主循环可以在需要的时候就从缓冲区里取到它。这并非难事，首先为下一个用户输入定义一个域，然后将 onKeyPressed 或 onTouchEvent 函数设为接到一个用户动作就放到那个域里，有这两步就够了。如果对于给定游戏的状态，这是一个合法的输入操作，那么所有输入需要在那一刻做的更新操作都已经定下来了，剩下的就让物理去关心怎样响应输入吧。

人工智能所做的类似于用户在决定下一个要“按”哪个按钮。学习怎样写人工智能程序超出了这篇文章的范围，但大体的意思是人工智能会按照用户的意图来按按钮。这些也有待物理去处理和响应吧。

物理可能是/不是真正的物理。对于动作类游戏来说，关键点是要考虑到上一次更新的时间、正在更新的当前时间、用户输入以及人工智能，并且决定它们朝着什么方向发展和是否会发生冲突。对于一个你可视化地抓取一些部件并滑动它们的游戏来说，物理就是这个游戏中滑动部件或者使之放入合适的位置的部分。对于一个小游戏来说，物理即使这个游戏中决定答案是错还是对的部分。你可能将其命名为其他东西，但每个游戏都有一个作

为游戏引擎的红肉部分（译者注：可能是主体部分的意思），在这篇文章里，我把这部分称为物理。

动画并非像在游戏里放入会动的 GIF 图片那样简单。你需要使得游戏能在恰当的时间画出每一帧。这并没有听起来那么困难。保留一些像 isDancing、danceFrame 和 lastDance FrameTime 那样的状态域，那样动画更新便能决定是否可以切换到下一帧去了。动画更新真正做的事就那么多。真正来显示动画的变化是由录像更新来处理的。

声音更新要处理触发声音、停止声音、音量变化以及音调变化。正常情况下写游戏的时候，声音更新会产生一些传往声音缓冲区的字节流，但是 Android 能够管理自己的声音，因而你的选择将是使用 SoundPool 或者 MediaPlayer。它们都需要小心处理以免出错，但你要知道，因为一些底层实现细节，小型、低比特率的声音文件将带来最佳的性能和稳定性。

录像更新要考虑游戏的状态、角色的位置、分数、状态等，并将一切画到屏幕上。如果使用主循环，你可能需要使用 SurfaceView，并做一个“推”绘制。对于其他视图，视图本身能够调用绘制操作，主循环不必处理。SurfaceView 每秒产生的帧数最多，最适合于一些有动画或屏幕上有运动部件的游戏。录像更新所要做的工作是获取游戏的状态，并及时地为这个状态绘制图像。其他的自动化操作最好由不同的更新任务来处理。

4. 3D 还是 2D

在开始写游戏之前，你要决定是做 3D 的还是 2D 的。2D 游戏有一个低得多的学习曲线，一般更容易获得良好的性能。3D 游戏需要更深厚的数学技能，并且如果你不注意的话会有性能问题产生。如果你打算画比方框和圆圈更复杂的图形，还需要会使用 3D Studio 和 Maya 那样的建模工具。Android 支持 OpenGL 用来 3D 编程，并且在 OpenGL 方面有很多很好的教程可供学习。

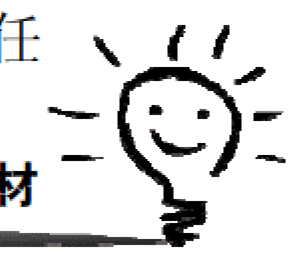
5. 建立简单、高质量的方法

上手时，要确保你整个游戏不要就用一个庞大而冗长的方法。如果你遵循我上面描述的主循环模式，这将相当简单。每个你写的方法应当完成一个非常特定的任务，并且它就应该无差错地那样做。举例来说，如果你需要洗一副纸牌，你应该写一个 shuffleCards 的方法，并且该方法就应该只做这一件事。

这是一个适用于任何软件开发的编码实践，但对于游戏开发来说这尤为重要。在一个有状态的、实时的系统里，调试将变得非常困难。所以你的方法要尽量的小，一般的经验法则是每个方法有且仅有一个目的（译者注：完成且仅完成一个功能）。如果你要为一个场景用编程方式画一个背景，你可能需要一个叫作 drawBackground 的方法。诸如此类的任务都能够很快完成，因而你可以按照搭积木的方法来开发你的游戏，而你也能够继续添加你要的功能，并且不会使这一切难以理解。

6. 最重要的是效率

性能是所有游戏的主要问题。我们的目标是使得游戏的反应越快越好，看起来越流畅越好。并且要将屏幕大小的位图画到主画布上，每一帧都是代价昂贵的。如何权衡对于达到最佳性能很有必要，确保管理好你的资源，使用技巧来以最少量的 CPU 资源完成你的任



务。如果性能不好的话，即使是最好的游戏玩起来也没劲。人们一般对于游戏卡或者响应慢几乎难以容忍。

7. 提示和技巧

看一下 SDK 中的示例 LunarLander。它使用 SurfaceView，这对于一个每秒需要处理最多帧的游戏来说是合适的。如果你要做 3D，示例中有 GLView 可以处理 3D 显示的很多初始化工作。对 LightRacer 来说，我不得不优化把所有东西都画出来这种方法，否则帧率将会大大地降低。我只在视图初始化的时候把背景画进一个位图里一次。路径放在它们自己的位图里，随着车手的前进而更新。这两个位图在每一帧里都被画进主画布中去，车手画在顶端，到最后会有一个爆炸。这种技术使得游戏运行在一个可以玩的程度。

如果适用的话，使得你的位图的大小精确到你打算画到屏幕上的大小，这也是个好的实践。这么做了之后还需要缩放，可以节省 CPU 资源。

在游戏中始终一致的位图配置（如 RGBA8888）。这将会通过减少不同格式之间转换的时间来节省图形库的 CPU 时间。

如果你决定开发 3D 游戏但没有 3D 方面的知识，你需要挑选一两本 3D 游戏编程方面的书并学习线性代数，至少要理解点积、叉积、向量、单元向量、法线、矩阵和变换。

声音文件要小而且低比特率。需要加载的越少，加载速度越快，游戏所需内存越少。

声音使用 OGG 文件，图片使用 PNG 文件。

确保释放所有媒体播放器，当 Activity 销毁时空出所有的资源。这能保证垃圾收集器清除所有东西，也能保证在两次游戏开始之间没有内存泄露。

加入 Android Googel 小组，寻求社区支持。这里有人可以在开发过程中给你帮助。

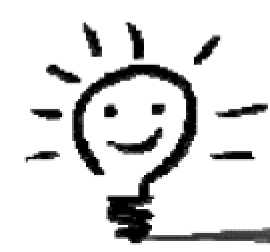
最重要的是，花时间测试再测试，确保每一小部分都如你所愿地工作。改善游戏是整个开发中最耗时、最困难的部分。如果你匆匆将其推向市场，很可能会使用户失望，你会感到你的努力都白费了。你不可能使所有人都喜欢你写的东西，但至少要尽量发布你最高质量的作品。

综合实训十一　微博随身之快速发布模块的实现

【问题情境】

前面已经介绍了个人中心 FunctionTabActivity 的开发，下面将介绍快速发布模块的实现。快速发布模块由 PublishActivity、PublishDiaryActivity、ShootActivity 及 UploadActivity 组成，其中 PublishActivity 起到了导航的作用。

在 PublishActivity 中包含了一个 ListView，该 ListView 中显示三个 Item：更新心情、发布日志和拍照上传。单击不同的 Item 将启动相应的 Activity 对话框。由于篇幅所限，本书将不再详述 PublishActivity 的具体开发细节。



【拓展知识】

1. 发布日志和更新心情功能的实现

下面将介绍发布日志和更新心情功能的开发。首先介绍发布日志功能的实现，在 PublishActivity 中单击“发布日志”，会启动 PublishDiaryActivity。

（1）PublishDiaryActivity 的开发

首先来看 PublishDiaryActivity 的开发，其代码如下。

```
1   package wyf.wpf;
2   import static wyf.wpf.ConstantUtil.SERVER_ADDRESS;
3   ...
4   import android.widget.Toast;
5   public class PublishDiaryActivity extends Activity{
6       MyConnector mc = null;
7       ProgressDialog pd = null;
8       String uno = null;
9       protected void onCreate(Bundle savedInstanceState) {
10      super.onCreate(savedInstanceState);
11      Intent intent = getIntent();
12      uno = intent.getStringExtra("uno");
13      setContentView(R.layout.publish_diary);
14      Button btnDiary = (Button)findViewById(R.id.btnDiary);//获得发布日志按钮
15      btnDiary.setOnClickListener(new View.OnClickListener() {
16                  public void onClick(View v) {
17                  publishDiary();                 //发表日志
18                  }
19          });
20          Button btnDiaryBack = (Button)findViewById (R.id.btnDiaryBack);
21          btnDiaryBack.setOnClickListener(new View.OnClickListener() {
22              public void onClick(View v) {
23                  finish();
24              }
25          });
26      }
27      public void publishDiary(){                 //方法：连接服务器，发表日志}
28      protected void onDestroy() {
29  }
```

☑ 第 9～23 行为重写的 onCreate 方法的代码，该方法的主要功能是设置当前显示的屏幕和为按钮添加 OnClickListener 监听器。

☑ 第 27 行为 publishDiary 方法的代码，该方法将在后面的步骤中介绍。

☑ 第 28 行为重写的 onDestroy 方法的代码，该方法将调用 MyConnector 对象的 sayBye 方法施放与服务器的连接。

上述代码中第 27 行省略了 publishDiary 方法的代码，该方法的功能是连接服务器将用户填写的日志信息公布，其代码如下。

```
public void publishDiary(){
        new Thread(){
            public void run(){
                Looper.prepare();
                EditText etTitle = (EditText) findViewById(R.id.etTitle);
                //获得日志标题 EditText 对象
                EditText etDiary = (EditText) findViewById(R.id.etDiary);
                //获得日志内容 EditText 对象
                String title = etTitle.getEditableText().toString().trim();    //获得日志标题
                String diary = etDiary.getEditableText().toString().trim(); //获得日志内容
                if(title.equals("") || diary.equals("")){     //如果标题或内容为空
                    Toast.makeText(PublishDiaryActivity.this, "请将日志的标题或内容填写完整",
                    Toast.LENGTH_LONG).show();
                    return;
                }
                try{
                    if(mc == null){                 //判断 MyConnector 是否为 null
                        mc = new MyConnector(SERVER_ADDRESS, SERVER_PORT);
                        //创建一个 Socket 连接
                    }
                String message = "<#NEW_DIARY#>" + title+"|"+diary+"|"+uno;
                mc.dout.writeUTF(message);              //发出消息
                String reply = mc.din.readUTF();        //接收消息
                pd.dismiss();
            if(reply.equals("<#DIARY_SUCCESS#>")){ //如果日志发布成功
                    pd.dismiss();//关闭进度对话框
                    Toast.makeText(PublishDiaryActivity.this, "日志发布成功，请在个人日志中查看",
                Toast.LENGTH_LONG).show();
                Looper.loop();
            }
            else if(reply.equals("<#DIARY_FAIL#>")){  //如果日志发布失败
                pd.dismiss();                          //关闭进度对话框
                Toast.makeText(PublishDiaryActivity.this, "日志发布失败，请稍候重试！",
                Toast.LENGTH_LONG).show();
                Looper.loop();//执行消息队列
            }
            }catch(Exception e){e.printStackTrace();}      //捕获打印异常
            Looper.myLooper().quit();                     //结束消息队列的循环
            }
    }.start();
}
```

- ☑ 第 5～12 行判断用户填写的日志信息是否完整，如果不完整，将会通过 Toast 提示用户。
- ☑ 第 18 行将用户填写的日志信息组装成带有消息头的字符串，第 19 行将字符串发送到服务器。
- ☑ 第 22～30 行为收到服务器反馈并进行判断的代码，并根据日志发布的成功或失败弹出不同的 Toast 提示。

（2）服务器的处理

用户在 PublishDiaryActivity 界面填写好要发布的新日志后，由 ServerAgent 负责接收消息并进行相应处理，该部分代码如下。

```
else if(msg.startWith("<#NEW_DIARY#>")){                 //消息为发布新日记
    msg = msg.substring(13);                              //获得消息内容
    String [] sa = msg.split("\\|");                      //获得字符串数组
    String result = DBUtil.writeNewDiary(sa[0],sa[1],sa[2]); //调用 DBUtil 类的方法发布日志
    String reply = "";
    if(result.equals(DIARY_SUCCESS)){                     //日志发布成功
        reply = "<#DEARY_SUCCESS#>";                      //设置返回字符串消息
    }
    else{                                                 //日志发布失败
        reply = "<#DIARY_NAME#>"};                        //设置返回字符串消息
    }
    dout.writeUTF(reply);                                 //向客户端发出回复消息
}
```

说明

上述代码首先从客户端发来的消息中去掉消息头，提取出日志的标题和内容，然后调用 DBUtil 类的 writeNewDiary 方法发布新的日志，最后根据 writeNewDiary 的返回值的不同向客户端反馈不同的消息。

更新心情模块的开发与发布新日志类似，只是更新心情以对话框的形式而不是 Activity 的形式呈现给用户，且该对话框属于 PublishActivity。由于篇幅所限，本书将不再详述该模块的代码。

2. 拍照上传界面的开发

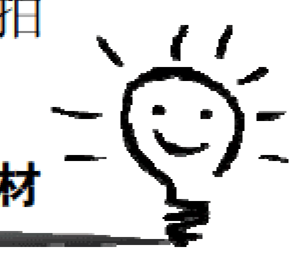
下面将介绍拍照上传模块的开发，该模块包括拍摄照片和上传照片两个部分，下面将分别介绍拍摄照片和上传照片功能的开发。

（1）拍摄照片功能的开发

拍摄照片功能由 ShootingActivity 实现。该 Activity 负责调用系统的照相机拍摄照片，本书不再详述。

（2）上传照片功能的开发

上传照片功能由 UploadActivity 实现，该 Activity 由 ShootingActivity 启动，负责将拍



摄的照片上传到服务器。接下来将按照如下步骤介绍 PublishDiaryActivity。

① 首先来看 PublishDiaryActivity 的代码框架，其代码如下。

```
package wyf.wpf;                                                    //声明包语句
import java.util.ArrayList;                                         //引入相关类
…//此处省略部分引入相关类的代码
import android.widget.Toast;                                        //引入相关类
public class UploadActivity extends Activity{
        MyConnector mc = null;                                      //声明 MyConnector 对象引用
        List<String []> albumList = new ArrayList<String []>();
        String uno = null;                                          //存放用户 ID
        byte [] data = null;                                        //存储图片的字节数组
        String newAlbum = null;                                     //存放新相册的名称
        String xid = null;                                          //要上传到的相册 ID
        String name = null;                                         //上传的照片名称
        String des = null;                                          //上传的照片描述
        ProgressDialog pd = null;                                   //声明进度对话框对象
        Spinner sp = null;                                          //Spinner 对象引用
        BaseAdapter baSpinner = new BaseAdapter(){
        //此处省略 baSpinner 的创建代码，将在随后补全};
        Handler myHandler = new Handler(){//创建用于处理 Handler 消息的 Handler 对象};
        protected void onCreate(Bundle savedInstanceState){         //重写 onCreate 方法
            super.onCreate(savedInstanceState);
            setContentView(R.layout.upload);                        //设置显示屏幕
            Intent intent = getIntent();                            //获得
            uno = intent.getStringExtra("uno");                     //读取用户 ID
            data = intent.getByteArrayExtra("data");                //读取字节数组
            getAlbumList();                                         //获得相册列表
            sp = (Spinner)findViewById(R.id.AlbumSpinner);          //获得 Spinner 对象
            sp.setAdapter(baSpinner);                               //设置 Spiner 的 Adapter
            Button btnNewAlbum = (Button)findViewById(R.id.btnNewAlbum);
            //获得添加新相册按钮
            …//此处省略为 btnNewAlbum 添加 OnClickListener 监听器的代码
            Button btnUpload = (Button)findViewById(R.id.btnConfirmUpload);//上传按钮
            …//此处省略为 btnUpload 添加 onClickListener 监听器的代码
            Button btnUploadBack = (Button)findViewById(R.id.btnUploadBack);
            btnUploadBack.setOnClickListener(new View.OnClickListener(){
            //为“返回”按钮添加监听器
                public void onClick(View v) {
                    UploadActivity.this.finish();                   //结束该 Activity 运行
                }
            });
    }
    public void getAlbumList(){                                     //方法：获得相册列表
    public void createNewAlbum(){//方法：与服务器通信，创建一个新相册}
    public void uploadMyPhoto(){//方法：向服务器上传照片}
    protected void onDestroy(){//重写 onDestroy 方法，释放 MyConnector 对象的资源}
    }
```

☑ 第 7 行声明了用于存放用户相册信息的 ArrayList，UploadActivity 类的 getAlbumList 方法负责连接服务器获取用户的相册信息。

☑ 第 8 行和第 9 行声明了用于存放用户 ID 和图片字节数组的成员变量 uno 和 data，这两个成员变量是由启动该 Activity 的 Activity 通过 Intent 传递过来的。

☑ 第 18～37 行为重写的 onCreate 方法。第 21 行调用 getIntent 方法获取启动该 Activity 的 Intent 对象，随即在代码第 22 行和 23 行通过该 Intent 获得用户的 ID 及在 ShootingActivity 部分拍摄的照片的字节数组。

☑ 第 28 行和第 30 行省略了为“创建新相册”和“确定上传”按钮添加监听器的代码。第 32～36 行为“返回”按钮添加了监听器。单击“返回”按钮将结束本 Activity 的执行。

② 上述代码第 16 行省略了创建新的 BaseAdapter 对象的代码，具体代码如下。

```
1   BaseAdapter baSpinner = new BaseAdapter() {                    //创建 BaseAdapter
2   public View getView(int position, View convertView, ViewGroup parent){//重写 getView 方法
3           TextView tv = new TextView(UploadActivity.this);       //创建 TextView
4           tv.setTextSize(18.5f);                                 //设置 TextView 字体大小
5           tv.setTextColor(R.color.character);                    //设置 TextView 字体颜色
6           String [] sa = albumList.get(position);                //获得本选中相册的信息
7           tv.setText(sa[1]);                                     //设置 TextView 显示的内容
8           return tv;
9           }
10          public long getItemId(int position) {                  //重写 getItemId 方法}
11    public Object getItem(int position) {                        //重写 getItem 方法}
12    public int getCount() {                                      //重写 getCount 方法}
13  }
```

☑ 开发一个新的 BaseAdapter 对象主要是重写其中的 getView 和 getCount 方法，getCount 方法用于返回要显示的内容的长度，而 getView 方法则决定每个要显示的 Item 的内容、布局方式及样式等。

☑ 第 2～9 行为重写的 getView 方法的代码，在该方法中创建一个 TextView 并设置了该控件的内容及样式并返回。第 12 行重写的 getCount 方法返回 albumList 的长度。

③ ①中代码第 28 行省略了为“创建新相册”按钮添加监听器的代码，这段代码如下。

```
btnNewAlbum.setOnClickListener(new View.OnClickListener() {   //为按钮添加监听器
        public void onClick(View v) {                         //重写 onClick 方法
        final View dialog_view=LayoutInflater.from(UploadActivity.this).inflate(R.layout. new_album, null);
        //获得对话框视图
        new AlertDialog.Builder(UploadActivity.this)          //创建一个对话框
        .setView(dialog_view)                                 //设置对话框要显示的 View
        .setPositiveButton(                                   //为对话框设置“确定”按钮
                R.string.btnOk,
        new DialogInterface.OnClickListener(){                //为“确定”按钮添加监听器
```



```
9                   @Override
10              public void onClick(DialogInterface dialog, int which){
11                  EditText et = (EditText)dialog_view.findViewById(R.id.etAlbumName);
12                  newAlbum = et.getEditableText().toString().trim();
13                  if(newAlbum.equals("")){                    //验证新相册的名称是否合法
14                      Toast.makeText(UploadActivity.this, "请输入新相册的名称", Toast.
                        LENGTH_LONG).show();
15                      return;
16                  }
17                      createNewAlbum();                       //连接服务器，创建新相册
18              }
19          }).show();                                          //显示对话框
20      }
21  }};
```

☑ 用户在上传图片界面中单击“创建相册”按钮后，会弹出一个让用户输入新相册名称的对话框。

☑ 第 3 行调用了 LayoutInflater 类的方法，获取了 new_album.xml 文件中的视图 View，该 View 对象将会作为对话框要显示的内容。

☑ 第 4～19 行创建并显示了一个 AlertDialog，第 6～19 行为对话框添加了一个 PositiveButton，并为其设置了监听器。代码第 10～18 行为按下对话框中的“确定”按钮后执行的代码，首先判断用户填写的新相册的名称是否合法，如果合法，则调用 createNewAlbum。

④ ①中代码第 30 行省略了“确定上传”按钮监听器的代码，这段省略的代码如下。

```
btnUpload.setOnClickListener(new View.OnClickListener(){  //为 btnUpload 添加监听器
    public void onClick(View v) {                          //单击“上传”按钮后执行的代码
        int position = sp.getSelectedItemPosition();       //获得相册 Spinner 被选中的位置
        if( position < 0){                                 //如果 Spinner 中无内容，返回-1
        Toast.makeText(UploadActivity.this, "您还没有选择上传相册！ ", Toast.LENGTH_
        LONG).show();
        //提示用户出错信息
            return;
        }
            xid = albumList.get(position)[0];              //获得要上传的相册的 ID
            EditText etName = (EditText)findViewById(R.id.etPhotoName);
            //获得照片名称的 EditText 控件
            EditText etDes = (EditText)findViewById(R.id.etPhotoDes);
            //获得照片描述的 EditText 控件
            name = etName.getEditableText().toString().trim(); //获得照片名称
            des = etDes.getEditableText().toString().trim();   //获得照片描述
            if(name.equals("") || des.equals("")){             //验证填写的内容是否完整
                    Toast.makeText(UploadActivity.this,
                    "请输入照片  名称", Toast.LENGTH_LONG).show();
```

```
17                  return;
18              }
19              pd = ProgressDialog.show(UploadActivity.this,
20              "请稍候", "正在上传图片...",true,true);          //显示进度
21              uploadMyPhoto();                                //调用方法上传图片
22              }
23      });
```

☑ 第 3～7 行判断用户是否还未创建过任何相册，如果相册列表为空，那么第 3 行代码执行后返回值为-1。

☑ 第 10～17 行为判断用户填写的新照片的名称和描述是否合法的代码，如果用户填写正确，则调用 uploadMyPhoto 方法进行图片上传。

3. UploadActivity 通信功能的开发

在前面的已经对 UploadActivity 界面的开发进行了简单的介绍，下面将讲解其通信功能的实现。UploadActivity 中主要通过三个方法来实现与服务器的通信。下面将分别介绍这三个方法的开发。

（1）getAlbumList 方法

getAlbumList 方法的功能是连接服务器获取指定用户的相册列表，该方法的代码如下。

```
public void getAlbumList(){                     //方法：获得相册列表
    new Thread(){                               //创建线程
    public void run(){                          //重写线程
        try{
            if(mc == null){
                mc = new MyConnector(SERVER_ADDRESS, SERVER_PORT);
            }
        mc.dout.writeUTF("<#GET_ALBUM_LIST#>"+uno);     //发出获取相册列表请求
            String reply = mc.din.readUTF();            //读取相册列表
            if(reply.equals("<#NO_ALBUM#>")){           //如果无相册
                Toast.makeText(UploadActivity.this,
            "您还没有创建任何相册！ ", Toast.LENGTH_LONG).show();
                albumList = new ArrayList<String []>();
                albumList.add(new String[]{null,"无相册"}); //添加到容器中
                return;
        }
        String albumArray [] = reply.split("\\$");      //切割字符串
        albumList = null;
        albumList = new ArrayList<String []>();
        for(String s:albumArray){
            String [] sa = s.split("\\|");              //切割字符串
            albumList.add(sa);                          //添加到容器中
            }
        }
        catch(Exception e){e.printStackTrace();}        //捕获并打印异常
```

```
26                  myHandler.sendEmptyMessage(0);              //发送消息
27              }
28              }.start();                                      //启动线程
29          }
```

☑ 第 8 行将用户的 ID 组装为带有消息头的字符串消息，并将消息发送到服务器。

☑ 第 10 行判断用户是否从未创建过相册，如果该用户从未创建过相册，则手动向相册信息列表添加一条显示信息。

☑ 第 17～23 行为当用户相册列表不为空时执行的代码，第 17 行执行完毕后，albumArray 数组中每个元素代表一个相册的信息。第 20～23 行对 albumArray 数组中的每个字符串进行切割，将相册信息中的各个部分分离出来组成字符串数组，并添加到 albumList 中。

ServerAgent 中处理该请求消息的代码如下。

```
else if(msg.startsWith("<#GET_ALBUM_LIST#>")){                  //消息为获取相册列表
        msg = msg.substring(18);                                //提取内容
        ArrayList<String []> albumList = DBUtil.getAlbumList(msg);
        if(albumList.size() == 0){                              //如果用户无相册
            dout.writeUTF("<#NO_ALBUM#>");                      //向客户端返回消息
        }
        else{                                                   //用户相册列表不为空
            StringBuilder sb = new StringBuilder();             //创建 StringBuilder 对象
            for(String [] sa:albumList){  //将 ArrayList 中的 String 数组转化为特定格式的字符串
                sb.append(sa[0]);                               //添加相册信息
                sb.append("|");                                 //添加分隔符
                sb.append(sa[1]);
                sb.append("|");                                 //添加分隔符
                sb.append(sa[2]);
                sb.append("$");                                 //添加分隔符
            }
            dout.writeUTF(sb.substring(0, sb.length()-1));//向客户端发送包含相册信息的字符串
        }
    }
```

说明

当收到头为"<#GET_ALBUM_LIST#>"的消息时，首先提取出要查询的相册编号，然后调用 DBUtil 类的 getAlbumList 方法获得相册信息列表，如果用户的相册列表不为空，则将相册信息组装成特定格式的字符串发回客户端。

（2）createNewAlbum 方法

createNewAlbum 方法的功能是连接服务器创建一个新的相册，该方法代码如下。

```
1   public void createNewAlbum(){                          //方法：与服务器交互，创建一个新相册
2       new Thread(){
3           public void run(){                             //重写 run 方法
4               Looper.prepare();                          //启动一个消息循环
5               try{
6                   if(mc == null){                        //判断 MyConnector 对象是否为空
7                       mc = new MyConnector(SERVER_ADDRESS, SERVER_PORT);
8                   }
9                   mc.dout.writeUTF("<#NEW_ALBUM#>"+newAlbum+"|"+uno);
                    //向服务器发出创建新相册的请求
10              String reply = mc.din.readUTF();           //接收服务器的消息
11              if(reply.equals("<#NEW_ALBUM_SUCCESS#>")){     //如果创建成功
12                  getAlbumList();                        //重新获得相册列表
13                  Toast.makeText(UploadActivity.this, "相册创建成功！", Toast.LENGTH_
                    LONG).show();
14                  Looper.loop();                         //执行本线程中的消息队列
15              }
16              else{
17                  Toast.makeText(UploadActivity.this, "相册创建失败，请重试！", Toast.
                    LENGTH_LONG).show();
18                  Looper.loop();
19              }
20          }catch(Exception e){e.printStackTrace();}
21          Looper.myLooper().quit();                      //在线程最后关闭消息队列
22          }
23      }.start();                                         //启动线程
24  }
```

☑ 第 9 行将用户填写的新相册名称及用户 ID 组装起来并添加消息头"<#NEW_ALBUM#>"，第 10 行将组装好的消息头字符串发出。

☑ 第 11～19 行对服务器反馈的消息进行判断，并根据相册创建成功与否显示相应的提示消息。

服务端 ServerAgent 类中处理该消息请求的代码如下。

```
else if(msg.startsWith("<#NEW_ALBUM#>")){                  //消息为创建新相册
    msg = msg.substring(13);                               //提取内容
    String [] sa = msg.split("\\|");                        //分割字符串
    int result = DBUtil.createAlbum(sa[0],sa[1]);          //调用 DBUtil 类的方法创建新相册
    if(result == 1){//创建成功
        dout.writeUTF("<#NEW_ALBUM_SUCCESS#>");            //发回创建成功消息
    }else{
        dout.writeUTF("<#NEW_ALBUM_FAIL#>");               //发回创建失败消息
        }
}
```






说明

ServerAgent 在收到头为"<#NEW_ALBUM#>"的消息时，首先从消息中提取出新相册名称和所属的用户 ID，然后调用 DBUtil 类的 createAlbum 进行创建，最后根据 createAlbum 方法的返回值向客户端发出相应的反馈消息。

（3）uploadMyPhoto 方法

uploadMyPhoto 方法的功能是向服务器上传一张照片到指定的相册中，其代码如下。

```
1   public void uploadMyPhoto(){                          //方法：向服务器上传照片
2           new Thread(){
3               public void run(){
4                   Looper.prepare();                     //为该线程初始化一个消息队列
5                   if(mc == null){                       //检查是否需要创建 MyConnector 对象
6                       mc = new MyConnector(SERVER_ADDRESS, SERVER_PORT);
7                   }
8                   try{
9                   String msg = "<#NEW_PHOTO#>"+name+"|"+des+"|"+xid;   //组织消息字符串
10                      mc.dout.writeUTF(msg);                 //发出消息
11                      mc.dout.writeInt(data.length);         //发出图片字节数组长度
12                      for(int i=0;i<data.length;i++){
13                          mc.dout.writeByte(data[i]);        //以字节为单位发送数据
14              }
15              mc.dout.flush();
16              String reply = mc.din.readUTF();               //接收回复
17              if(reply.equals("<#NEW_PHOTO_SUCCESS#>")){     //上传成功
18                  pd.dismiss();
19                  Toast.makeText(UploadActivity.this, "照片上传成功！ ", Toast.LENGTH_LONG).
                    show();
20                  Looper.loop();
21              }
22              else{//上传失败
23                  pd.dismiss();
24                  Toast.makeText(UploadActivity.this, "照片上传失败！ ", Toast.LENGTH_LONG).
                    show();
25                  Looper.loop();
26              }
27          }catch(Exception e){e.printStackTrace();}//捕获并打印异常
28          }
29      }.start();
30  }
```

☑ 第 9 行将用户填写的相片信息机相册 ID 信息组装并添加消息头"<#NEW_PHOTO#>"组成消息字符串，代码第 10 行将消息字符串发出。在发出消息字符串之后，继续发送图片字节数组的长度，然后将图片字节数组发送到服务器。

☑ 第 17～24 行对服务器反馈的消息进行判断，并根据图片上传的成功与否提示相应



的信息。

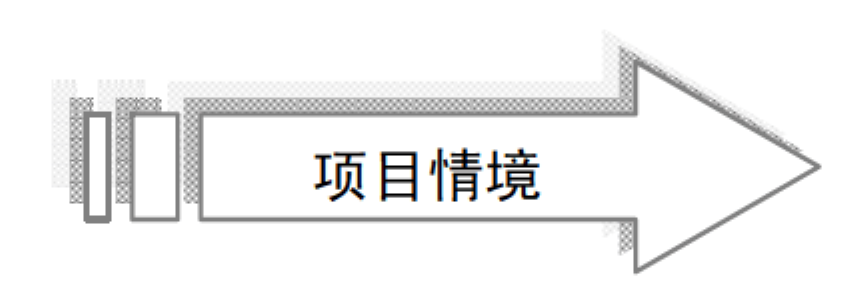
服务端 ServerAgent 类中处理该消息请求的代码如下。

```
else if(msg.startsWith("<#NEW_PHOTO#>")){                              //消息为上传照片
        msg = msg.substring(13);                                        //提取内容
        String [] sa = msg.split("\\|");                                //分隔字符串
        int size = din.readInt();                                       //读取图片大小
        byte [] buf = new byte[size];                                   //创建字节数组
                for(int i=0;i<size;i++){
                        buf[i] = din.readByte();                        //读取图片字节数据
                }
        int result = DBUtil.insertPhotoFromAndroid(buf,sa[0],sa[1], sa[2]);  //插入图片
        if(result == 1){
                dout.writeUTF("<#NEW_PHOTO_SUCCESS#>");                 //返回上传成功消息
        }else{
                dout.writeUTF("<#NEW_PHOTO_FAIL#>");                    //返回上传失败的消息
        }
}
```

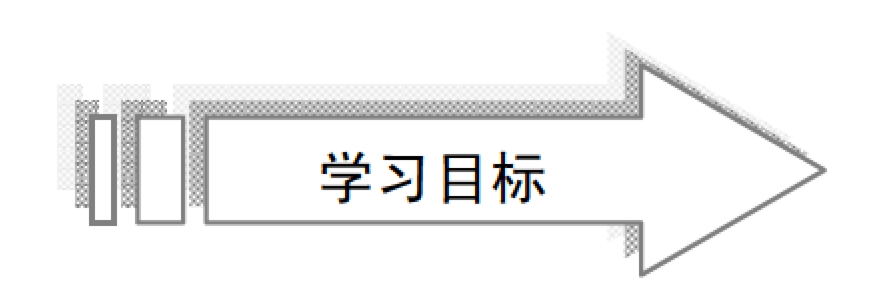
说明

ServerAgent 收到头为"<#NEW_PHOTO#>"的消息时，首先提取其中的照片名称、描述和所属相册等信息，然后接收客户端发来的图片字节数组长度，并创建相应长度的字节数组。在读取到图片字节数组后，调用 DBUtil 类的 insertPhotoFromAndroid 方法向数据库插入图片数据，最后根据 insertPhotoFromAndroid 方法的返回值向客户端返回相应的成功或失败消息。

项目十二
地图中可遇实体模块的开发



游戏中英雄每走完指定骰子数的地图格子，都将检测当前位置是否与地图的可遇实体发生相遇。本项目简单介绍可遇实体对象的开发，其中涉及到的类有 MyDrawable、MyMeetableDrawable 以及继承自 MyMeetableDrawable 的各个子类。



- 掌握 MyDrawable 类的开发方法
- 掌握 MyMeetableDrawable 类的开发方法
- 掌握 ForestDrawable 类的开发方法
- 掌握可遇实体对象的调用流程

任务 29　MyDrawable 类的开发

【任务情境】

MyDrawable 是 MyMeetableDrawable 的父类，游戏中不可遇的地图元素如道路、花等均是 MyDrawable 类的对象。

【相关知识】

在本游戏中，MyDrawable 的宽度和高度可以是地图图元大小（31 像素）的任意倍数，如代表稻田的大小为 62×62，占地图中 4 个格子。

当 MyDrawable 的大小超过 31×31 时，就需要为其指定参考点，在绘制 MyDrawable 时将根据其参考点以及在地图中所占的行和列进行绘制。在本游戏中，参考点用相对于 MyDrawable 左下角的行和列来指定，图 12-1 中说明了如何基于参考点来绘制 MyDrawable。

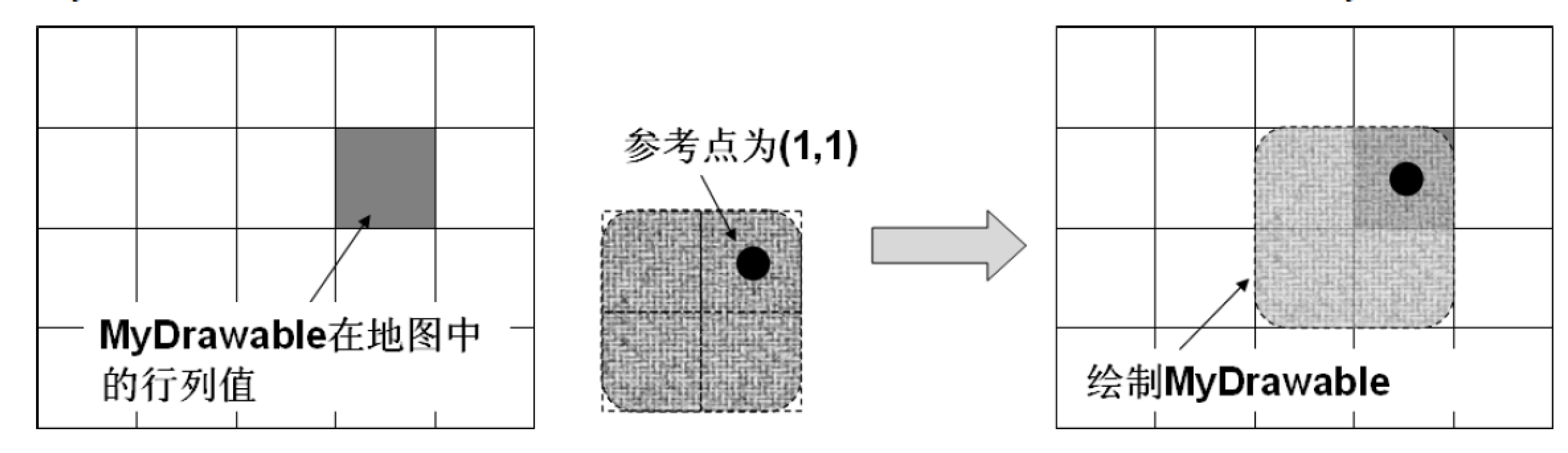


图 12-1　基于参考点来绘制 MyDrawable

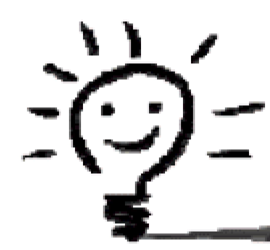
在图 12-1 中，要绘制的 MyDrawable 在地图中占 2 行 2 列，其中定位点位于右上角的那个格子，相对于左下角的坐标（1,1）。在绘制 MyDrawable 时，将参考点所在的格子对到 MyDrawable 在地图中所占的行和列即可。

同时，由于 MyDrawable 的大小可能超过一个地图格子的尺寸，在表示其可通过情况时，不可以用 0 或 1 来简单概括。因为有些 MyDrawable 可能部分位置可通过，其他位置不可通过，这时需要将 MyDrawable 内部所有不可通过的点记录下来。如图 12-1 中的 MyDrawable，如果其上边两个图元不可通过，下面两个可通过，则其不可通过点为（1,0）和（1,1），同样是相对于 MyDrawable 左下角的坐标。

下面来介绍 MyDrawable 类的开发，其代码框架如下。

```
package wyf.ytl;                                          //声明包语句
import static wyf.ytl.ConstantUtil.TILE_SIZE;             //引入相关类
import java.io.Externalizable;                            //引入相关类
import java.io.IOException;                               //引入相关类
```

```
import java.io.ObjectInput;                         //引入相关类
import java.io.ObjectOutput;                        //引入相关类
import android.graphics.Bitmap;                     //引入相关类
import android.graphics.Canvas;                     //引入相关类
/*该类中封装了一个地图图元的信息，每个 MyDrawable 类都在地图上占有一个格子。该类中包含图
片引用，图片宽高，图片位置（row，col），定位点坐标，不可通过矩阵。该类中还包含是否可遇的标志
位*/
public class MyDrawable implements Externalizable{
    private static final long serialVersionUID = 9191440096790116821L;//持久化版本序列号
    Bitmap bmpSelf;                                 //自己图片的引用
    int width;                                      //图元的宽度
    int height;                                     //图元的高度
    int col;                                        //在大地图中所在的列
    int row;                                        //在大地图中所在的行
/*以上两行声明了用于记录 MyDrawable 对象在地图中位置的成员变量*/
    int refCol;                 //定位参考点在本 MyDrawable 中所占的列，以左下角为原点
    int refRow;                 //定位参考点在本 MyDrawable 中所占的行，以左下角为原点
/*以上两行声明了用于记录参考点在 MyDrawable 内部位置的成员变量*/
    int [][] noThrough;                             //不可通过矩阵
/*记录 MyDrawable 自身的通过情况的不可通过矩阵，由于不可通过点可能不止一个，所以将所有不
可通过点存放到一个数组中*/
    boolean meetable;                               //是否可以遇到
/*声明的 meetable 为该 MyDrawable 可遇与否的标志位，true 表示可遇，false 表示不可遇*/
    public void writeExternal(ObjectOutput out) throws IOException {}
    public void readExternal(ObjectInput in) throws IOException,
                ClassNotFoundException {}
/*以上两行为对 Externalizable 接口中 writeExternal 和 readExternal 方法的实现*/
    public MyDrawable(){}                           //无参构造器
/*MyDrawable 的无参构造器，这些都是服务于游戏存档和读取模块的，在此将不详述*/
    public MyDrawable(Bitmap bmpSelf,boolean meetable,int width,int height,int col,int row,int refCol,
int refRow,int [][] noThrough){}
/*MyDrawable 为构造器，在该构造器中将 MyDrawable 的主要成员变量初始化*/
/*drawSelf 方法，该方法首先通过将要绘制到屏幕上的行和列以及参考点的位置计算出 MyDrawable
左上角的 x 和 y 坐标，然后减去各自的偏移量进行绘制*/
    public void drawSelf(Canvas canvas,int screenRow, int screenCol,int offsetX,int offsetY){
    //方法：绘制自己
        int x = (screenCol-refCol)*TILE_SIZE;       //求出自己所拥有的块数中左上角块的 x 坐标
        int y = screenRow*TILE_SIZE+(refRow+1)*TILE_SIZE-height;
        //求出自己所拥有的块数中左上角块的 y 坐标
        canvas.drawBitmap(bmpSelf, x-offsetX, y-offsetY, null);//根据自己的左上角的 x、y 坐标画出自己
    }}
```



任务 30　MyMeetableDrawable 类的开发

【任务情境】

MyMeetableDrawable 为抽象类，其子类对象代表地图中的可遇实体，要想让 MyMeetableDrawable 具有可遇的特性，除了要继承 MyDrawable 外，还需要实现 View.OnTouchListener 接口。

【相关知识】

游戏中当英雄与可遇实体发生相遇时，可遇实体对象将会用自己替换掉 GameView 的 View.OnTouchListener 监听器以执行特定的业务逻辑。MyMeetableDrawable 类的代码框架如下。

```
package wyf.ytl;                                //声明包语句
import static wyf.ytl.ConstantUtil.*;           //引入相关类
import java.io.Externalizable;                  //引入相关类
import java.io.IOException;                     //引入相关类
import java.io.ObjectInput;                     //引入相关类
import java.io.ObjectOutput;                    //引入相关类
import android.graphics.Bitmap;                 //引入相关类
import android.graphics.BitmapFactory;          //引入相关类
import android.graphics.Canvas;                 //引入相关类
import android.graphics.Paint;                  //引入相关类
import android.view.MotionEvent;                //引入相关类
import android.view.View;                       //引入相关类
public abstract class MyMeetableDrawable extends MyDrawable implements View.OnTouchListener,
Externalizable {
    int [][] meetableMatrix;                    //可遇矩阵，相对于本 MyDrawable 所占的块数
/*声明了用于记录可遇对象中的可遇点的二维数组，同 MyDrawable 的不可通过矩阵一样，可遇实体
中的可遇点也需要额外记录，meetableMatrix 中记录的同样是可遇部分相对于可遇实体左下角的坐标*/
    Bitmap bmpDialogBack;                       //对话框背景图片
    Bitmap bmpDialogButton;                     //对话框的按钮背景图片
/*以上两行声明了两个 Bitmap 对象的引用，分别代表对话框的背景图片和对话框按钮的背景图片，
当英雄遇到可遇实体时，是通过对话框的方式来交互的*/
    Hero tempHero;                              //英雄的引用，用于备份数据防止污染
    public void writeExternal(ObjectOutput out) throws IOException {}
    public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {}
    //构造器
    public MyMeetableDrawable(){}
    public MyMeetableDrawable()
    public abstract void drawDialog(Canvas canvas,Hero hero);    //在游戏屏幕上绘制对话框的方法
```

```
/*抽象方法 drawDialog，因为不同的可遇实体所执行的逻辑不同，所以要绘制的对话框也不尽相同，各个子类需要独立实现该方法*/
public void drawString(Canvas canvas,String string){}        //绘制给定的字符串到对话框上
/*drawString 方法，该方法负责将接收到的字符串对象绘制到指定的 Canvas 对象中，MyMeetable Drawable 子类在实现 drawDialog 方法时会调用到 drawString 方法*/
    public boolean onTouch(View arg0, MotionEvent arg1) {//实现 View.OnTouchListener 接口的方法
/*对 View.OnTouchListener 接口的 onTouch 方法的实现，在 MyMeetableDrawable 类中对此只提供了空实现，该方法将在子类中进行具体实现*/
}
```

任务 31　ForestDrawable 类的开发

【任务情境】

前面对 MyDrawable 和 MyMeetableDrawable 类进行了介绍，游戏中的真正与英雄相遇的是 MyMeetableDrawable 的子类对象，如 ForestDrawable、PalaceDrawable 等。

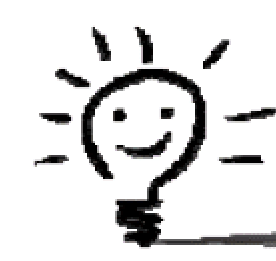
【相关知识】

不同的可遇实体执行不同的业务逻辑，但其都是通过重写 MyMeetableDrawable 类的抽象方法 drawDialog 和实现 View.OnTouchListener 接口的 onTouch 方法来完成与玩家的交互性。下面将以 ForestDrawable 为例来说明地图可遇实体的开发过程，其步骤如下。

（1）ForestDrawable 在地图中显示为森林，当英雄与 ForestDrawable 相遇时可以消耗一定的体力值来伐木，伐木会获得一定数额的金钱。为实现这些功能，在 ForestDrawable 类中需要另外声明一些成员变量，这些成员变量的代码如下。

```
String dialogMessage[] = {  //对话框的提示信息，第一个是可以狩猎，第二个是不可以狩猎
/*声明了一个 String 数组，数组中每个元素代表不同情况下对话框需要显示的文本信息*/
"此处树林阴翳，林中必然有许多参天大树。是否伐木？预计消耗体力 xx，收获 yy 金。"
/*字符串中包含了“xx”和“yy”，这两处将会在绘制对话框之前被计算出的消耗体力值和收获金钱数所替换掉*/
"此处是个伐木的好地方，怎奈人困马乏，体力不足，只好改日再来！"
};
int result;                 //记录英雄可以获得的收益
/*声明了用于存储英雄可获得的收益的成员变量，在 ForestDrawable 类中，该成员变量代表收获的金钱数*/
int status;                 //状态，0：显示是否伐木的选择对话框，1：显示不能伐木的提示对话框
/*声明的 status 代表对话框应处于的状态，0 表示向玩家显示是否伐木的对话框，1 表示向玩家显示不能伐木的对话框*/
```

（2）接下来需要实现父类的抽象方法 drawDialog，在 ForestDrawable 类中，drawDialog 方法代码如下。



```
public void drawDialog(Canvas canvas,Hero hero) {          //方法：绘制对话框
    String showString = null;                               //需要显示到对话框中的字符串
    tempHero = hero;                                        //先画背景
    canvas.drawBitmap(bmpDialogBack, 0, DIALOG_START_Y, null);
    Skill skill = hero.heroSkill.get(LUMBER);               //获得英雄的伐木技能
    result = skill.calculateResult();                       //计算英雄的收益
/*调用了英雄伐木技能的计算收益的方法，该方法将在随后的内容中介绍*/
/*通过判断伐木技能的 calculateResult 方法的返回值来确定对话框的状态*/
    if(result == -1){                                       //体力值不够了
        status = 1;                                         //设置对话框状态
        showString = dialogMessage[status];                 //设置对话框显示的文本信息
    }else{                                                  //如果体力值够
        status = 0;                                         //设置对话框状态
        showString = dialogMessage[status];                 //设置对话框显示的文本信息
        showString = showString.replaceAll("xx", skill.strengthCost+"");//替换消耗体力的字符串
        showString = showString.replaceFirst("yy", result+"");//替换掉预计收获金钱数的字符串
    }
    drawString(canvas, showString);                         //根据对话框状态绘制相应的信息
    //绘制“确定”按钮
/*将确定按钮绘制到对话框中，首先将按钮背景图片绘制出来，然后创建一个 Paint 画笔对象并对其
进行设置，最后将按钮文本绘制到对话框中*/
    canvas.drawBitmap(bmpDialogButton, DIALOG_BTN_START_X, DIALOG_BTN_START_Y, null);
    Paint paint = new Paint();                                          //创建画笔对象
    paint.setARGB(255, 42, 48, 103);                                    //设置画笔颜色
    paint.setAntiAlias(true);                                           //设置抗锯齿
    paint.setTypeface(Typeface.create((Typeface)null,Typeface.ITALIC)); //设置字体
    paint.setTextSize(18);                                              //设置字号
    canvas.drawText("确定",                                             //绘制“确定”文字
            DIALOG_BTN_START_X+DIALOG_BTN_WORD_LEFT,
            DIALOG_BTN_START_Y+DIALOG_WORD_SIZE+DIALOG_BTN_WORD_UP,
            paint
            );
/*判断是否需要绘制“取消”按钮，因为对于状态 1（即不可伐木的状态）是不需要绘制“取消”按
钮的*/
    //绘制“取消”按钮
    if(status == 0){                                  //查看是否需要画第二个“取消”按钮
        canvas.drawBitmap(bmpDialogButton, DIALOG_BTN_START_X
                    +DIALOG_BTN_SPAN, DIALOG_BTN_START_Y, null);
        canvas.drawText("取消",                       //绘制“取消”文字
            DIALOG_BTN_START_X+DIALOG_BTN_SPAN+DIALOG_BTN_WORD_LEFT,
            DIALOG_BTN_START_Y+DIALOG_WORD_SIZE+DIALOG_BTN_WORD_UP,
            paint);
    }}
*/
```

（3）将对话框绘制出来后，当玩家单击按钮后，需要进行相应的处理，这些处理工作

放在 onTouch 方法中进行，该方法来自接口 View.OnTouchListener.onTouch 方法，其代码如下。

```
public boolean onTouch(View v, MotionEvent event) {    //对 onTouch 方法的实现
        int x = (int)event.getX();                      //获得单击位置的 x 坐标
        int y = (int)event.getY();                      //获得单击位置的 y 坐标
        if(event.getAction() == MotionEvent.ACTION_DOWN){
/*当玩家单击“确定”按钮时的处理代码：首先判断此时对话框的状态，如果为 0（即玩家确认伐木），则调用伐木技能的 useSkill 方法；如果为 1（玩家不能伐木时按下确认），则什么也不做。代码最后一行为 recoverGame 方法，该方法用于恢复游戏状态*/
            if(x>DIALOG_BTN_START_X && x<DIALOG_BTN_START_X+DIALOG_BTN_
            WIDTH && y>DIALOG_BTN_START_Y && y<DIALOG_BTN_START_Y+
            DIALOG_BTN_HEIGHT){                         //单击的是“确定”按钮
                switch(status){                         //根据状态来处理“确定”按钮按下事件
                case 0:                                 //有木可伐时玩家确认伐木
                    tempHero.heroSkill.get(LUMBER).useSkill(result);
                    //调用技能类的 useSkill 方法
                    break;
                case 1:                                 //无木可伐时玩家确认放弃
                    break;
                }
                recoverGame();                          //恢复游戏状态
/*当玩家单击“取消”按钮时的处理代码：玩家按下“取消”按钮的情况只有一种，即可以伐木时选择不伐木。代码倒数第二行也调用了 recoverGame 方法来恢复游戏状态*/
            }else if(x>DIALOG_BTN_START_X+DIALOG_BTN_SPAN
            && x<DIALOG_BTN_START_X+DIALOG_BTN_SPAN+DIALOG_BTN_WIDTH
            && y>DIALOG_BTN_START_Y
            && y<DIALOG_BTN_START_Y+DIALOG_BTN_HEIGHT){
            //判断单击的是否是“取消”按钮
                recoverGame();
            }}
        return true;
    }
```

任务 32　可遇实体对象的调用流程

【任务情境】

在前面以 ForestDrawable 为例介绍了 MyMeetableDrawable 子类对象的开发，下面将介绍可遇实体对象在游戏中的调用流程。



【相关知识】

英雄从检查是否与可遇实体相遇到结束与可遇实体的交互之间要经过如下流程。

- ☑ 调用 HeroGoThread 类的 checkIfMeet 方法判断是否相遇。
- ☑ 如果英雄与某个可遇实体相遇，用可遇实体对象的监听方法替换掉 GameView 的 View.onTouchListener 监听器，并进行设置让 GameView 调用可遇实体对象的 drawDialog 方法。
- ☑ 可遇实体对象与玩家交互完毕后，调用可遇实体对象的 recoverGame 方法恢复游戏。

接下来介绍上述的流程在代码中的具体实现，其步骤如下。

（1）HeroGoThread 类的 checkIfMeet 方法的开发，其代码如下。

```
public boolean checkIfMeet(){                    //方法：检查英雄是否与可遇实体对象相遇
MyMeetableDrawable mmd = gv.meetableChecker.check(hero);
/*调用了 GameView 类中 MeetableLayer 的 check 方法，将英雄停留位置的左右两边的可遇实体对象返回
*/
if(mmd != null && mmd!=gv.previousDrawable){
/*如果碰到了可遇物，且该可遇物并不是上一次那个则对返回值进行判断，如果其不为空，且不是上次遇到的可遇实体，则执行代码后续 5 行的内容*/
/*首先设置 GameView 的 OnTouchListener 为可遇实体对象 mmd，然后将 mmd 赋值给 GameView 中的 currentDrawable 和 previousDrawable。currentDrawable 用于记录当前的可遇实体对象，previousDrawable 用于记录上次的可遇实体对象*/
        gv.setOnTouchListener(mmd);          //设置 GameView 的 OnTouchListener
        gv.currentDrawable = mmd;            //设置 GameView 的当前可遇实体
        gv.previousDrawable = mmd;           //记录为前一个，以防下次走一步遇到了同样的那个
            if(gv.activity.isEnvironmentSound){gv.playSound(1, 0);}       //如有需要，播放声音
            return true;
            }
            gv.previousDrawable = mmd;       //将 mmd 记录到 GameView 的成员变量中
            return false;
        }
```

说明

> 在 GameView 中设置 currentDrawable 和 previousDrawable 的目的是防止两次遇到的可遇实体相同。假设游戏中的森林所占的行列数较多，英雄两次掷骰子走动后仍然与该可遇实体相遇，如果不做如上设置，将会触发两次伐木时间，这并不符合游戏逻辑。

（2）在 GameView 中绘制可遇实体的对话框，onDraw 方法为 GameView 的绘制方法。可遇实体对象的 drawDialog 也是在 onDraw 方法中调用的，该代码如下。

```
if(currentDrawable != null){                              //判断当前的可遇实体对象是否为 null
        currentDrawable.drawDialog(canvas, hero);         //调用可遇实体对象的 drawDialog 方法
}
```



（3）对话框被显示，监听器被替换后，玩家就可以与可遇实体对象进行交换了，交互完毕之后通过调用可遇实体对象的 recoverGame 方法来恢复游戏，该方法代码如下。

```
public void recoverGame(){                                    //方法：返还监听器，恢复游戏状态
    tempHero.father.setOnTouchListener(tempHero.father);     //返还监听器
    tempHero.father.setCurrentDrawable(null);                //置空记录引用的变量
    tempHero.father.setStatus(0);                            //重新设置 GameView 为待命状态
    tempHero.father.gvt.setChanging(true);                   //骰子转起来
}
```



说明

在 recoverGame 方法中首先将监听器重新设置为 GameView 自身，然后将 GameView 类的 currentDrawable 引用置空并将游戏状态设置为 0（待命态），最后通过设置 GameView Thread 类的标志位让骰子重新变换起来。

【项目小结】

本项目介绍的是可遇实体对象的开发方法，其中涉及到的类有 MyDrawable、My MeetableDrawable 以及继承自 MyMeetableDrawable 的各个子类，其用处在于检测当前位置是否与地图的可遇实体发生相遇，这一功能在类似游戏中都需要实现，读者需要掌握在心。

综合实训十二　微博随身之查看联系人模块的开发

【问题情境】

下面将介绍口袋微博 Android 端查看联系人模块的开发，该模块主要用到了 Contacts Activity 类。

【拓展知识】

1. ContactsActivity 界面的开发

接下来将介绍 ContactsActivity 界面的开发，ContactsActivity 界面中只包含一个 List View，因此该 Activity 的主要功能是为 ListView 提供要显示的数据，并确定显示内容的风格样式，ContactsActivity 的代码如下。

```
package wyf.wpf;                          //声明包语句
import java.util.ArrayList;               //引入相关类
...//此处省略部分引入相关类的代码
import android.widget.AdapterView.OnItemClickListener;
public class ContactsActivity extends Activity{
```



```
        String uno = null;                          //记录当前用户的 ID
        int type = -1;                              //为 0 表示显示好友列表，为 1 表示显示访客列表
        Bitmap [] headList = null;                  //存放头像的数组
        ArrayList<String[]> infoList = null;
    /*存放联系人信息的列表如果是好友则为 ID、姓名、E-mail、状态、头像。若为访客则为 ID、姓名、
日期、头像*/
        MyConnector mc = null;                      //网络连接器对象
        ListView lv = null;                         //ListView 对象引用
        String [] messageHead = {"<#FRIEND_LIST#>","<#VISITOR_LIST#>"};
        //消息头字符串  数组
        BaseAdapter baContacts=null;                //ListView 的 BaseAdapter 对象引用
        Handler myHandler = new Handler(){          //创建 Handler 对象
        @Override
        public void handleMessage(Message msg) {            //重写 handleMessage 方法
        switch(msg.what){
        case 0:
            lv.setAdapter(baContacts);                      //设置 ListView 的 adaper
            break;
        }
        super.handleMessage(msg);
        }
    };
    protected void onCreate(Bundle savedInstanceState) {    //重写 onCreate 方法
        super.onCreate(savedInstanceState);
        Intent intent = getIntent();                        //获得启动该 Activity 的 Intent 对象
        uno = intent.getStringExtra("uno");                 //获得当前用户的 ID
        type = intent.getIntExtra("type", -1);              //获取类型，显示好友还是访客
        if(type == 0){                                      //好友列表
            baContacts = new BaseAdapter() {
            //此处省略创建用于显示好友的 BaseAdapter 的代码};
        else if(type == 1){                                 //访客列表
            baContacts = new BaseAdapter()
            {//此处省略创建用于显示好友的 BaseAdapter 的代码};
        }
        setContentView(R.layout.contacts);                  //设置当前屏幕
        lv = (ListView)findViewById(R.id.listFriend);       //获得 ListView 对象的引用
        getContact();                                       //查询服务器，获得联系人列表
        lv.setOnItemClickListener(new OnItemClickListener() {
        //为 ListView 添加监听器
            public void onItemClick(AdapterView<?> parent, View v, int position,long id) {
            //重写 onIteClick 方法
            Intent intent = new Intent(ContactsActivity.this,HomePageActivity.class);
            intent.putExtra("uno", infoList.get(position)[0]);      //设置 Intent 的 Extra 字段
            intent.putExtra("visitor", uno);                        //设置 Intent 的 Extra 字段
            startActivity(intent);                                  //启动 Activity
            }
        });
```

```
46    }
47    public void getContact(){              //方法：获取联系人列表
48    protected void onDestroy() {           //重写 onDestroy 方法，释放 MyConnector
49    }
```

- ☑ 第 6 行声明了用于记录用户 ID 成员变量，第 7 行声明了用于标识显示类别的成员变量，如果为 0，表示显示好友列表，为 1 表示显示访客列表。这两个成员变量都将从启动该 Activity 的 Intent 对象中获取。
- ☑ 第 8 行声明了用于存放联系人头像的 Bitmap 数组，第 9 行声明了用于存放联系人信息的 ArrayList，这两个成员变量将在 getContact 方法中被赋值。
- ☑ 第 12 行声明了一个存放消息头的 String 数组，在与服务器通信时将根据 type 值的不同选取不同的消息头。
- ☑ 第 25～46 行为重写的 onCreate 方法，该方法首先根据 type 值的不同创建不同的 BaseAdapter 对象，然后根据 type 值的不同获取不同的联系人列表。
- ☑ 第 38～45 行为 ListView 添加了 OnItemClickListener 监听器，在重写的 onItemClick 方法中创建了一个 Intent 对象并使用该 Intent 启动 HomePageActivity 访问联系人的微博主页。

2. ContactsActivity 通信功能的开发

ContactsActivity 中负责与服务器通信的是 getContact 方法，该方法的功能为查询服务器获取联系人的头像及其他信息，其代码如下。

```
public void getContact(){                      //方法：获取联系人列表
    new Thread(){
        public void run(){
            try{
                mc = new MyConnector(SERVER_ADDRESS, SERVER_PORT);
                //创建 My Connector 对象
                mc.dout.writeUTF(messageHead[type]+uno); //向服务器发出请求
                int size = mc.din.readInt();             //读取列表的长度
                headList = null;
                infoList = null;
                headList = new Bitmap[size];             //初始化好友头像列表
                infoList = new ArrayList<String []>(size);   //初始化好友信息列表
                for(int i=0;i<size;i++){          //循环，获取每个好友的信息和头像
                    String fInfo = mc.din.readUTF();      //读取好友信息
                    String [] sa = fInfo.split("\\|");    //分割字符串
                    infoList.add(sa);             //将好友信息添加到相应的列表中
                    int headSize = mc.din.readInt();      //读取头像大小
                    byte[] buf = new byte[headSize];      //创建缓冲区
                    mc.din.read(buf);                     //读取头像信息
                    headList[i] = BitmapFactory.decodeByteArray
                    (buf, 0, headSize);                   //创建 Bitmap
                }
```

```
21                  }catch(Exception e){e.printStackTrace();}          //打印并捕获异常
22                  myHandler.sendEmptyMessage(0);
23              }
24          }.start();
25      }
```

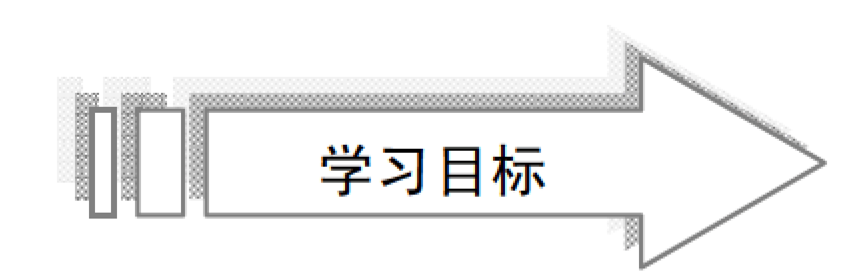
☑ 第 6 行根据成员变量 type 值的不同向服务器发出不同消息头的查询请求。第 7 行接收服务器传来的联系人个数，并据此创建 headList 数组和 infoList 列表。

☑ 第 12～20 行为获取指定个数联系人的信息的代码，其中获取头像的方法是接收服务器传来的图片文件的字节数组并调用 BitmapFactory 的 decodeByteArray 方法创建 Bitmap 对象。

项目十三
英雄技能模块的开发



在前面介绍 ForestDrawable 类的开发时，曾经提到了英雄的技能并调用了技能对象的方法。本项目介绍英雄的技能模块的开发。该模块涉及到的类有 Skill、FarmingSkill、SuiXin BuSkill 等，其中 Skill 为所有技能类的基类。



- 学习 Skill 类的开发框架
- 掌握继承自 Skill 类的技能子类对象的开发方法
- 以随心步为例掌握特殊技能的开发方法

任务 33　Skill 类的开发

【任务情境】

Skill 类为抽象类，每个子类在继承 Skill 类时需要重写父类的抽象方法。

【相关知识】

Skill 类的代码框架如下。

```
package wyf.ytl;                                   //声明包语句
import java.io.Serializable;                       //引入相关类
/*该类为技能类，每个该类对象代表一个技能，其中包含技能的熟练度。每次释放技能会消耗一定的
体力，技能类中有一个 use 方法，每次释放技能都会调用，用于计算所得和增加熟练度等操作*/
public abstract class Skill implements Serializable{
    private static final long serialVersionUID = 5878577464788782434L;    //持久化版本号
    int id;//技能 ID，唯一标识一个技能
/*声明了 Skill 对象的 ID，每种技能都具有唯一的 ID，其值取自 ConstantUtil 中的常量，如 LUMBER
代表伐木技能的 ID 等*/
    String name;                                   //技能的名字
    int proficiencyLevel;                          //技能熟练度
/*记录技能的熟练度的成员变量，随着技能熟练度的提高，英雄获得的收益也会增加*/
    int strengthCost;                              //每次释放消耗的体力
    int basicEarning;                              //收入的单位，用于计算英雄的总收入
    int tempProficiency;                           //临时熟练度，到一定程度会增加技能熟练度
/*记录临时熟练度的成员变量，当临时熟练度达到代码下一行定义的 proficiencyToUpgrade 时，技能
会升级*/
    int proficiencyToUpgrade;                      //技能熟练度升级需要的临时熟练度
    int skillType;                   //技能类型，包括 0：生活技能、1：战斗技能、2：普通技能
    Hero hero;                                     //所属英雄的引用
    public Skill(){}                               //无参构造器
    public Skill(int id,String name,int basicEarning,int skillType,Hero hero){}  //有参构造器
    //Skill 类的有参构造器，在该构造器中对主要的成员变量进行初始化
    //方法：每次使用前调用，根据返回英雄应得的收益，如果无收益则返回-1;
    public abstract int calculateResult();
    //方法：释放技能，该消耗的消耗，扫尾工作
    public abstract void useSkill(int skillEarning);
…//此处省略成员变量的 get 和 set 方法
    //以上两行为抽象成员方法 calculateResult 和 useSkill 的声明
}
public static final int LUMBER = 0;                //代表伐木技能，作为技能 HashMap 的键
```




```
public static final int FISHING = 1;             //代表打鱼技能，作为技能 HashMap 的键
public static final int FARMING = 2;             //代表农耕技能，作为技能 HashMap 的键
public static final int MINING = 3;              //代表采矿技能，作为技能 HashMap 的键
public static final int SUI_XIN_BU = 4;          //代表随心步
public static final int HUI_TOU_SHI_AN = 5;      //代表回头是岸
public static final int WU_ZHONG_SHENG_YOU = 6;  //代表无中生有
```

任务 34　LumberSkill 类的开发

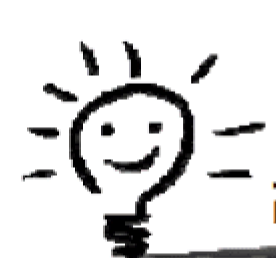
【任务情境】

本任务将以 LumberSkill 为例，介绍如何开发继承自 Skill 类的技能子类对象。

【相关知识】

LumberSkill 对象代表伐木技能，实现 LumberSkill 主要是对其父类的抽象方法进行重写，其代码如下。

```
/*calculateResult 方法的代码，在该方法中首先判断英雄当前的体力值是否足够施放技能，如果不够
返回 0，否则调用 GameFormula 的静态方法来计算英雄通过这次伐木能够收货的金钱*/
public int calculateResult(){                       //方法：计算英雄收益，若无收益返回 0
    if(hero.strength < strengthCost){               //如果英雄的体力不够使用本次技能
        return -1;
    }else{                                          //利用公式计算英雄收益
        int result = GameFormula.getSkillEearning(proficiencyLevel, basicEarning);
        return result;
    }}
/*useSkill 方法的代码，在该方法中将修改与该技能有关的英雄数据，如金钱增加、体力值减少等，
同时还增加技能的临时熟练度*/
public void useSkill(int skillEarning){             //方法：使用技能
hero.setTotalMoney(hero.getTotalMoney() + skillEarning);    //英雄金钱增加
    hero.setStrength(hero.getStrength() - strengthCost);    //英雄体力减少
    tempProficiency += PROFICIENCY_INCREMENT;       //英雄临时熟练度增加
/*对技能的临时熟练度进行判断，如果其值达到升级的上限且还没有升到最大等级，则对技能进行升
级操作*/
if(tempProficiency == proficiencyToUpgrade && proficiencyLevel<SKILL_LEVEL_MAX){
//可以熟练度升级了
        proficiencyLevel += 1;                      //熟练度等级加 1
        tempProficiency = 0;                        //临时熟练度置零
        strengthCost -= STRENGTH_COST_DECREMENT;           //消耗的体力值减少
        proficiencyToUpgrade+=PROFICIENCY_UPGRADE_SPAN; //下一级升级上限增加
    }}
```





说明

伐木技能的使用流程可以参考之前对 ForestDrawable 类中的 drawDialog 和 onTouch 方法代码的介绍，其他生活类技能的使用流程也与此类似。

任务 35　SuiXinBuSkill 类的开发

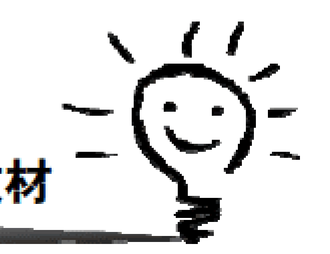
【任务情境】

在任务 34 中介绍了伐木技能 LumberSkill 的开发，伐木技能属于生活技能，游戏中英雄还具有其他的特殊技能，如随心步技能可以让英雄前进指定的地图格子数，回头是岸技能可以让英雄调转方向，无中生有技能可以让英雄凭空获得一些士兵和金钱等。

【相关知识】

本任务将以随心步技能为例来说明特殊技能的开发，特殊技能同样继承自 Skill 类，需要对父类的抽象方法进行实现，SuiXinBuSkill 类的代码如下。

```
/*calculateResult 方法的代码，由于随心步技能不会产生收益，所以该方法在此的功能是判断英雄能否施放，返回 0 表示能够施放该技能，返回-1 表示不能施放*/
public int calculateResult(){                    //方法：计算技能收益，在此为判断是否能够使用
    if(hero.strength < strengthCost){            //判断英雄的体力值够不够
        return -1;                               //返回-1 表示不能够使用该技能
    }
    return 0;                                    //返回 0 表示能够使用该技能
}
/*useSkill 方法的代码，施放随心步技能后首先减少英雄的体力值，然后获得 GameView 类中用于记录随心步步数的成员变量的值，最后调用 Hero 类的 startToGo 方法激活 HeroGoThread 使英雄开始移动，同时还需要将 GameView 的状态设置为 1（英雄移动状态）*/
public void useSkill(int skillEarning){                      //方法：施放技能
    hero.setStrength(hero.getStrength() - strengthCost);     //减少体力
    int steps = hero.father.suiXinBu;                        //获取要走几步
    hero.startToGo(steps);                                   //英雄开始走
    hero.father.setStatus(1);                                //设置 GameView 状态
}
```



【项目小结】

本项目介绍了英雄的技能模块的开发方法，包括 Skill、FarmingSkill、SuiXinBuSkill 等，其中 Skill 为所有技能类的技能，该方法实现英雄的技能并调用技能对象，希望读者能够熟练掌握，类似游戏的英雄技能都会用到相似的方法。

📖 小知识十八：关于成功上线一款应用的建议

很多人都曾有过关于手机应用很好的点子，我们也相信大家能够开发出一款足够吸引眼球、超过所有同类的独特、具有革新性的手机应用，或许现在你就拥有伟大的想法，甚至你都已经完成了应用的设计和开发架构。但不是所有的应用都要上线。

为了一个应用能够顺利上线和吸引用户，应该根据下面的上线指南来做准备。

1．应用上线前一个月

（1）制定目标。为了确定你能成功，你需要设定一些可量化的目标，尽可能的简单。比如刚开始时，把目标定在主动安装率、评论数量和平均分数排名上。

（2）制作一个视频。视频是表现你应用的界面和功能最有效并最具表现力的手段之一。认真考虑如何制作一个能很好的展示应用特点和工作流程的视频，这并不是指你要花钱让别人来做，用智能手机屏幕模拟器、免费的 Simfinger 和 iShowU HD （高级版要 60 美元），你就可以自己做一个简单但有专业效果的视频 Demo，并且还能把应用最酷的功能向用户展示出来。

（3）决定你接下来要怎么走。如果你要发布一款 Android 应用的话，不要在每个商店都注册，集中在一个用户群上，在一个应用商店专注促销活动，尽可能多地将潜在用户转化成真正用户，这样你的应用就会得到很高的评分、排名和曝光量。分销渠道越少，你的应用的更新和数据跟踪就越简单。先从官方的智能手机市场开始，建立了一定的知名度后再扩展到其他平台。

2．应用上线前一周

（1）接触媒体。理想的情况是：报道者和博主如果能在应用上线时写出有关你的应用的文章，那你就需要提前让他们知道。在应用上线前选择一些少量的曝光，确保每个人都能知道应用上线的准确日期和时间。准备好足够的时间去回答可能出现的问题，确保你在应用上线前一天或当天有时间去媒体那里，因为他们可能会需要一些额外的信息。记住你要在应用上线那天接触到所有之前研究过的报道者。

（2）提供一个预览版。最好能给媒体提供你应用的预览版，在正式上线之前如果能让测评人试用一下你的应用，那他们的报道将会更有深度和前瞻性。

（3）组织并完成你的多媒体资料。你的多媒体资料越多（包括插图、Logo、屏幕截图和视频等），你的文章和市场效果看起来就越漂亮、越有现场感，对于读者来说也就越有趣、越有话题感。确保每一个你接触的报道者都能在应用上线之前获得所有的多媒体资料。



3. 应用上线当天

（1）接触你媒体名单上的每一个人。这样的话在应用上线当天你会得到大量的正面报道。为了提高知名度，接触额外媒体名单中相关的报道者和博主，让他们知道你的应用今天上线了，确保把你应用的详细信息发给那些大的应用评论网站，以及其他你认为会感兴趣的媒体。

（2）向你现存的用户推荐一下。如果你在一个新的平台上发布应用，一定要鼓励一下你已有的粉丝们帮你传播一下。一般来说现有的用户群在新应用上线当天传播你的应用会收到意想不到的效果，所以要通过博客、社交媒体、小广播等尽可能的媒体来传播给你的现有用户，同时一定要让你的用户帮你传播或转发一下，因为一个朋友效应的活动会得到更多的响应。

（3）懂得利用社交。在应用上线的一整天最好在 Twitter 或在 Facebook 上多宣布几次。利用发布当天的新鲜感，通过一些有奖竞答的转发活动来推广你的 Twitter 活动，同时不要忘记随时查看所有的社交媒体版块，回复一句“谢谢”或者在适当的时候转发一下使用者的话。

4. 应用后续的宣传跟踪

在激动的上线之后，该如何保持这种增长势头，不让你的应用在快速增长后迅速下滑呢？

（1）支付广告费。如果你的应用很火，适当地支付宣传活动费用会明显保持和延长当初上线效应以及吸引用户、媒体注意力的那种势头。但一定要快，确保活动在应用上线后的那几天都是有效的。

（2）不要放弃媒体。在你的应用刚刚上线后，不要忘记去感谢那些有兴趣和有时间来参加报道的人；在你继续开发或在你的应用上添加新功能时，确保他们会跟踪报道。

（3）与其他的应用开发者合作。如果能集成其他的应用，那无疑是一种既能增加应用新功能又能获得之前应用的用户的好方法。在选择合适的应用集成合作伙伴时，首先想想对你的用户来说最有价值的是什么，不要纯粹为了用户群而选择合作伙伴——因为这常常不是早期推广中最重要的。如果你和集成的合作伙伴的应用都是实用且有趣的，你们会获得成倍增长。

（4）继续激励你的用户。要在社交媒体上表现活跃：询问用户有哪些问题并及时回复他们的疑问、想法和反馈。如果你在新的更新中加入了他们的建议，一定要感谢他们的贡献。

综合实训十三　微博随身之日志管理模块的实现

【问题情境】

本实训将介绍口袋微博中日志管理模块的开发，该模块主要为用户提供查看、编辑和删除日志的功能，主要由 MyDiaryActivity 和 ModifyDiaryActivity 来实现。



【拓展知识】

1. 查看日志功能的开发

下面将介绍查看日志功能模块的开发，该模块由 MyDiaryActivity 实现，其功能是将用户的日志以 ListView 的形式显示出来，其代码如下。

```
package wyf.wpf;                                    //声明包语句
import android.app.Activity;                        //引入相关类
…//此处省略部分引入相关类的代码
import android.widget.ListView;
public class MyDiaryActivity extends Activity{
    MyConnector mc = null;
    ArrayList<String []> diaryList = new ArrayList<String []>();   //存放日志信息的 ArrayList
    ListView lvDiary = null;                        //声明 ListView 对象
    int positionToDelete = -1;                      //记录要删除日志在 ListView 中的位置
    String uno = null;//记录用户 ID
    Handler myHandler = new Handler(){              //创建自定义的 Handler 对象
    public void handleMessage(Message msg){         //重写 handleMessage 方法
            switch(msg.what){
            case 0:
                lvDiary.setAdapter(ba);             //设置 ListView 的 adapter
                break;
            }
            super.handleMessage(msg);
        }
    };
    BaseAdapter ba = new BaseAdapter() {//此处省略创建 BaseAdapter 对象的代码};
    View.OnClickListener listenerToEdit = new View.OnClickListener() {
    //创建 OnClickListener 监听器};
    View.OnClickListener listenerToDelete = new View.OnClickListener() {
    //创建 OnClick Listener 监听器};
    protected void onCreate(Bundle savedInstanceState) {    //重写 onCreate 方法
        super.onCreate(savedInstanceState);
        Intent intent = getIntent();                        //获得启动该 Activity 的 Intent
        uno = intent.getStringExtra("uno");                 //读取 Intent 中的 Extra 字段值
        setContentView(R.layout.diary);                     //设置当前屏幕
        lvDiary = (ListView)findViewById(R.id.lvDiary);     //获得 ListView
        getDiaryList();                                     //获得日志列表
    }
    public void deleteDiary(){//方法：删除指定日志}
    public void getDiaryList(){//方法：获取日志列表}
    protected void onDestroy() {}
}
```

- ☑ 第 11 行创建的 Handler 对象主要用于接收来自 getDiaryList 方法中的 Handler 消息，因为在非主线程中无法执行 ListView 的 setAdapter 方法，故采用了 Handler 消息传递机制。
- ☑ 第 21 行省略了创建 BaseAdapter 的代码，读者可自行查阅。
- ☑ 第 22 行和第 23 行省略了声明两个 OnClickListener 监听器的代码，在个人日志列表中，所有的“编辑”按钮都将添加 listenerToEdit 监听器，单击后跳转到编辑日志界面；而所有的“删除”按钮都将添加 listenerToDelete 监听器，单击后提示是否确认删除日志。
- ☑ 第 24～31 行为重写的 onCreate 方法，该方法的主要功能是设置当前屏幕并调用 getDiaryList 方法获取个人日志列表。

查看日志功能中与服务器进行通信的主要是 deleteDiary 和 getDiaryList 方法，这两个方法分别负责删除指定日志和获取个人日志列表，其代码如下。

```
public void deleteDiary(){                              //方法：删除指定日志
    new Thread(){
        public void run(){
            Looper.prepare();                           //开启一个消息循环
            try{
                if(mc == null){                         //检查是否需要创建 MyConnector 对象
                    mc = new MyConnector(SERVER_ADDRESS, SERVER_PORT);
                }
                String rid = diaryList.get(positionToDelete)[0];   //获得要删除的日志的编号
                String msg = "<#DELETE_DIARY#>"+rid;   //组织消息字符串
                mc.dout.writeUTF(msg);                  //发出消息
                String reply = mc.din.readUTF();        //读取返回信息
                if(reply.equals("<#DELETE_DIARY_SUCCESS#>")){   //删除成功
                Toast.makeText(MyDiaryActivity.this, "删除日志成功！ ", Toast.LENGTH_
                LONG). show();
                    getDiaryList();                     //刷新日志列表
                    Looper.loop();                      //执行消息队列
                }
                else{                                   //删除失败
                Toast.makeText(MyDiaryActivity.this, "删除失败，请重试！ ", Toast.LENG
                TH_LONG).show();
                    Looper.loop();                      //执行消息队列
                }
            }catch(Exception e){e.printStackTrace();}   //捕获并打印异常
            Looper.myLooper().quit();                   //结束消息队列
        }
    }.start();
}
public void getDiaryList(){                             //方法：获取日志列表
    new Thread(){
```

```
28          public void run(){
29              try{
30                  mc = new MyConnector(SERVER_ADDRESS, SERVER_PORT);
31                  mc.dout.writeUTF("<#GET_DIARY#>"+uno+"|"+"1");//组织和发出消息字符串
32                  int size = mc.din.readInt();                //读取日志的长度
33                  diaryList = null;
34                  diaryList = new ArrayList<String []>();      //创建 diaryList
35                  for(int i=0;i<size;i++){                      //循环接收日志信息
36                      String diaryInfo = mc.din.readUTF();     //读取日志信息
37                      String [] sa = diaryInfo.split("\\|");
38                      diaryList.add(sa);                       //将日志信息添加到列表中
39                  }
40                  myHandler.sendEmptyMessage(0);               //发出 Handler 消息
41              }catch(Exception e){   e.printStackTrace();}     //捕获并打印异常
42          }
43      }.start();
44  }
```

- ☑ 第 1～26 行为 deleteDiary 方法的代码，该方法中首先根据 positionToDelete 成员变量的值取得要删除的日志的编号，然后把日志编号组装消息字符串发给服务器，最后接收服务器的反馈消息并输出相应信息给用户。
- ☑ 第 26～44 行为 getDiaryList 方法的代码，该方法首先组装并发出用于查询的消息字符串。根据服务器反馈回来的日志的个数，循环读取每个日志的信息，将这些信息切割后添加到 diaryList 列表中。

2. 编辑日志功能的开发

边界日志功能由 ModifyDiaryActivity 来实现，该 Activity 的功能是获取指定日志的标题和内容，并将其显示到可编辑的 EditText 控件中供用户编辑。其实现方式与发表日志功能比较类似，本书由于篇幅所限，将不对该 Activity 进行详细介绍。

📖 小知识十九：移动应用发展——回顾 2011、展望 2012

近几年，智能手机和平板电脑的销售势头日益强劲，可谓是一年火过一年。2011 年更是不同寻常的一年，在这一年中，智能手机和平板电脑的销量首次超过了台式电脑和传统的笔记本电脑的销量，与此相伴的是移动应用的爆发式增长，人们花在应用上的时间也越来越多。目前，苹果 iOS 应用商店的应用下载量已突破 180 亿，而谷歌 Android 市场的应用下载量也已突破 100 亿大关。现在就让我们来共同回顾移动应用在 2011 年的发展概况，共同展望应用在 2012 年的发展趋势。

1. 2011 移动应用发展回顾

图 13-1 是人们在应用上和网页上花费时间的对比表。



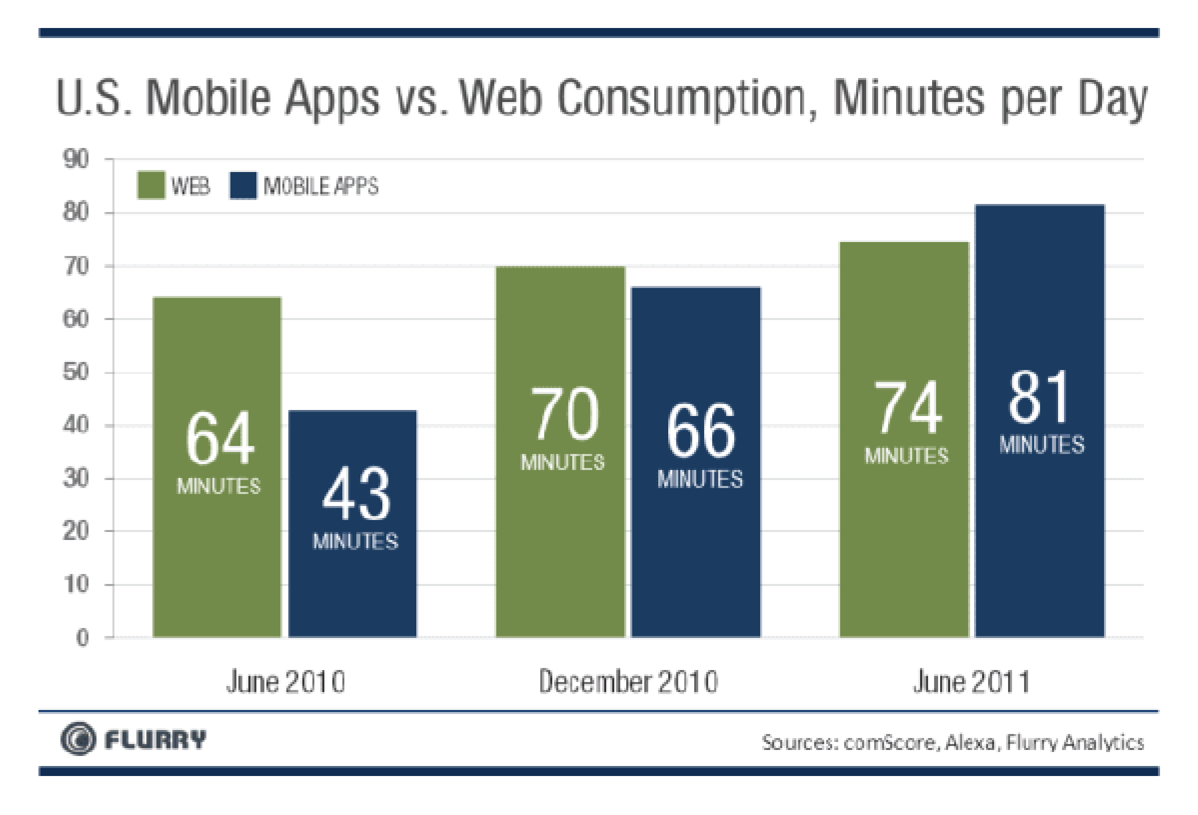


图 13-1 人们在应用上和在网页上花费时间对比

Flurry 发布的数据显示，在 2011 年 6 月份，人们平均每天花在应用上的时间为 81 分钟，而浏览网页的时间为 74 分钟，花在应用上的时间首次超过了浏览网页的时间。而在 2010 年的 6 月份，人们每天平均花在应用上的时间仅为 43 分钟，浏览网页的时间为 64 分钟。

人们在应用上花的时间超过了浏览网页的时间。

（1）iOS 平台上的应用开发者的收入要比 Android 应用开发者多得多

现在，在 Android 平台上，下载量最多的应用中的三分之二都是免费的，这就意味着 Android 设备用户购买应用的动机就非常弱，在这个方面，应用开发者的感受应该是最为深刻的。Flurry 发布的一项数据显示，Android 应用开发者的收入仅为 iOS 应用开发者的 24%。

（2）应用正以燎原之势席卷全球

在应用下载量不断增加的情况下，应用下载在全球分布形势也发生了巨大变化。2011 年 1 月份，美国的应用下载量占到全球的 55%，到 2011 年 10 月份，这个数字下降到 47%。与此同时，很多其他国家的应用下载量实现了爆发式增长，中国 2011 年的应用下载量较 2010 年增长了 870%，阿根廷增长了 527%。

（3）谷歌依然没能解决 Android 设备操作系统的升级问题

由于谷歌过于频繁地升级移动操作系统，这就使得很多 Android 设备不能得到及时持续的系统升级，这对运营商和开发商而言都是一个不小的问题。谷歌也承认了这个问题的存在，并许诺将出台一套相关规范来缓解这个问题，但这个问题并没有得到缓解。

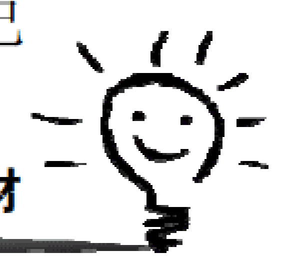
2. 2012 移动应用发展展望

（1）对语音控制技术的整合

苹果虽然并不是第一家在手机中添加语音控制技术的公司，但不可否认的是，iPhone 4S 里的语音助手 Siri 使得这项技术为众人熟知。在未来，苹果 Siri 的竞争对手会越来越多，据报道，谷歌正在开发一款对抗 Siri 的 Android 应用 Majel。此外，语音技术公司 Nuance 不久前刚收购竞争对手 Vlingo，意欲在这个领域有所作为。Nuance 旗下的输入法应用 Swype 也新增了语音输入功能。

（2）除移动支付领域外，NFC 的应用范围会更加广泛

很长一段时间以来，NFC 移动支付备受青睐。在 2012 年，人们依然看好 NFC，但已



经不仅仅局限于移动支付领域了，NFC 在卖家奖励忠实用户等方面发挥的作用也越来越大。此外，NFC 在明年也将慢慢与社交类应用和交换联系方式的应用进行整合。

（3）更多的应用商店将会出现

目前，在苹果 iOS 应用商店和谷歌的 Android 市场广受欢迎的背后，很多问题也开始凸显，在茫茫的应用海洋中，用户很难找到真正的优质应用，开发者要想得到关注也不那么容易。因此，类似亚马逊应用商店的其他品牌应用商店可能会陆续出现，百思买和 EA 都很有可能推出自己的应用商店。

项目十四

游戏提示模块的开发



在前面介绍 HeroBackDataThread 时曾经提到过，游戏中发生粮草危机、敌军突袭等事件时，需要提示玩家进行相应处理，这里就使用到了游戏提示模块的功能。本项目介绍游戏提示模块的开发，其中涉及到的类主要有 GameAlert 及其子类 FoodAlert、WarAlert、PlainAlert 以及 GameOverAlert。

学习目标

- 掌握所有游戏提示类的父类（GameAlert 类）的开发方法
- 掌握游戏提示对象的开发方法
- 掌握特定游戏提示的开发方法，包括 FoodAlert 类的代码框架、父类的抽象方法 drawDialog 的具体实现、onTouch 方法对单击事件的处理方法
- 掌握 GameAlert 在游戏中的用法

任务 36　GameAlert 类的开发

【任务情境】

游戏提示模块与可遇实体模块的开发比较类似。

【相关知识】

在该模块中，抽象类 GameAlert 为所有游戏提示类的父类，其代码如下。

```
package wyf.ytl;                                    //声明包语句
/*该类为抽象类，主要定义的动作是游戏中随时会弹出的提示信息，如大军来袭、粮草危机、科研成功等*/
import static wyf.ytl.ConstantUtil.*;               //引入相关类
import android.graphics.Bitmap;                     //引入相关类
import android.graphics.Canvas;                     //引入相关类
import android.graphics.Paint;                      //引入相关类
import android.graphics.Typeface;                   //引入相关类
import android.view.View;                           //引入相关类
public abstract class GameAlert implements View.OnTouchListener{
    Bitmap bmpDialogBack;                           //对话框背景
    Bitmap bmpDialogButton;                         //对话框按钮
/*以上两行声明了游戏提示中所用到的图像对象引用，游戏中的提示也是通过对话框的形式显示到屏
幕上的*/
    GameView gameView;                              //GameView 对象引用
    public GameAlert(GameView gameView,Bitmap bmpDialogBack,Bitmap bmpDialogButton){
        this.gameView = gameView;
        this.bmpDialogBack = bmpDialogBack;
        this.bmpDialogButton = bmpDialogButton;
    }
    /*抽象方法：用于获得监听后绘制对话框*/
    public abstract void drawDialog(Canvas canvas);
/*抽象方法 drawDialog，子类除了要实现该方法外还需要实现 View.OnTouchListener 接口的 onTouch 方法*/
    /*方法：绘制给定的字符串到对话框上*/
    public void drawString(Canvas canvas,String string){}
/*在对话框上绘制文本信息的方法，该方法与 MyMeetableDrawable 类中的 drawString 方法是一样的*/
```

任务 37　PlainAlert 类的开发

【任务情境】

前面的任务介绍了 GameAlert 类，下面将以 PlainAlert 类为例来介绍游戏提示对象的开发。



【相关知识】

PlainAlert 是最简单的一种 GameAlert，其功能是向玩家显示提示信息和一个确定按钮，玩家按下确定按钮后回到正常游戏的状态。PlainAlert 类的代码如下。

```
package wyf.ytl;//声明包语句
/*该类继承自抽象类 GameAlert，主要负责最简单的界面显示*/
import static wyf.ytl.ConstantUtil.*;                //引入相关类
import android.graphics.Bitmap;                      //引入相关类
import android.graphics.Canvas;                      //引入相关类
import android.graphics.Paint;                       //引入相关类
import android.graphics.Typeface;                    //引入相关类
import android.view.MotionEvent;                     //引入相关类
import android.view.View;                            //引入相关类
public class PlainAlert extends GameAlert{
    String alertMessage;                             //要显示的消息，需要设置才行
    /*声明 alertMessage 变量，该变量存储要显示的提示信息，将在代码下一行的构造器中被初始化*/
    //构造器
    public PlainAlert(GameView gameView,String alertMessage,Bitmap bmpDialogBack,Bitmap bmp
DialogButton){
        super(gameView,bmpDialogBack, bmpDialogButton);
        this.alertMessage = alertMessage;
    }
/*对父类抽象方法 drawDialog 方法的具体实现，由于 PlainAlert 只是用来提示信息，所以只需要绘制一个“确定”按钮即可*/
    public void drawDialog(Canvas canvas) {          //对父类抽象方法 drawDialog 的实现
        //绘制背景
        canvas.drawBitmap(bmpDialogBack, 0, DIALOG_START_Y, null);
        drawString(canvas, alertMessage);            //绘制“确定”按钮
        canvas.drawBitmap(bmpDialogButton, DIALOG_BTN_START_X, DIALOG_BTN_START_
Y, null);
        Paint paint = new Paint();                   //创建画笔对象
        paint.setARGB(255, 42, 48, 103);             //设置画笔颜色
        paint.setAntiAlias(true);                    //设置抗锯齿
    paint.setTypeface(Typeface.create((Typeface)null,Typeface.ITALIC));     //设置字体
        paint.setTextSize(18);                       //设置字号
        canvas.drawText("确定",                      //绘制“确定”按钮
                DIALOG_BTN_START_X+DIALOG_BTN_WORD_LEFT,
                DIALOG_BTN_ START_Y +DIALOG_WORD_SIZE+DIALOG_BTN_WORD_UP,
                paint);
    }
    //方法：实现 View.OnTouchListener 接口的方法
/*对 View.OnTouchListener 接口方法 onTouch 的实现，在玩家单击“确定”按钮后，通过执行最后三
```

```
行的代码来恢复游戏*/
        public boolean onTouch(View view, MotionEvent event) {
            int x = (int)event.getX();                    //获得单击处的 x 坐标
            int y = (int)event.getY();                    //获得单击处的 y 坐标
            if(event.getAction() == MotionEvent.ACTION_DOWN){
                if(x>DIALOG_BTN_START_X && x<DIALOG_BTN_START_X+DIALOG_BTN_
                WIDTH && y>DIALOG_BTN_START_Y && y<DIALOG_BTN_START_Y+
                DIALOG_BTN_HEIGHT){                        //单击“确定”按钮
                    gameView.setStatus(0);                 //设置游戏状态为 0（待命态）
                    gameView.setOnTouchListener(gameView); //将监听器重新设置为 GameView
                    gameView.currentGameAlert = null;      //置空当前游戏提示
                }
            }
            return true;
        }
    }
```

下面来介绍如何在程序中使用 PlainAlert，在游戏中如果需要显示提示信息，就需要创建一个 PlainAlert 对象，然后将其引入赋值给 GameView 类的成员变量 currentGameAlert。在 GameView 的 onDraw 方法中，将通过以下的代码绘制 PlainAlert。

```
    if(currentGameAlert != null && currentDrawable == null){  //判断是否需要绘制游戏提示
            currentGameAlert.drawDialog(canvas);              //绘制对话框
            setOnTouchListener(currentGameAlert);             //设置 GameView 的监听器
    }
```


说明

> 在游戏可遇实体和游戏提示都需要显示及替换 GameView 的监听器，本游戏人为规定可遇实体的优先级比游戏提示的高。因此在判断是否需要绘制游戏提示时，首先要保证记录当前可遇实体对象的 currentDrawable 引用为 null。

任务 38　FoodAlert 类的开发

【任务情境】

在游戏提示模块中除了 PlainAlert 可以在多种场合使用显示提示信息外，其他的 GameAlert 子类都只负责特定的提示信息的显示，如 FoodAlert 用在发生粮草危机的场合、WarAlert 用在敌军来袭的场合、GameOverAlert 用在英雄破产游戏结束的场合等。



【相关知识】

本任务将以 FoodAlert 为例介绍特定游戏提示的开发。其步骤如下。

（1）首先开发出 FoodAlert 类的代码框架，其代码如下。

```
package wyf.ytl;                                //声明包语句
import static wyf.ytl.ConstantUtil.*;           //引入相关类
import android.graphics.Bitmap;                 //引入相关类
import android.graphics.Canvas;                 //引入相关类
import android.graphics.Paint;                  //引入相关类
import android.view.MotionEvent;                //引入相关类
import android.view.View;                       //引入相关类
public class FoodAlert extends GameAlert{
    CityDrawable city;                          //记录是哪个城池粮草危机
    String alertMessage = "刚才 xx 城守将派人来报,说粮草已经不多了,是否划拨粮草?";
    //构造器
    public FoodAlert(GameView gameView,CityDrawable city,Bitmap bmpDialogBack,Bitmap bmp
DialogButton){}
    public void drawDialog(Canvas canvas){}     //对父类抽象方法的实现
    public boolean onTouch(View v, MotionEvent event){}
//对接口 View.OnTouchListener 接口方法的实现
}
```

说明

代码第 9 行声明了用于记录粮草危机城市的 CityDrawable 对象引用，该成员变量将在第 11 行的构造器中被初始化。代码第 10 行为将要显示的提示信息，该字符串中的“xx”部分将会在显示之前被替换为发生粮草危机的城市名称。

（2）接下来将对父类的抽象方法 drawDialog 进行具体实现，其代码如下。

```
public void drawDialog(Canvas canvas) {                                    //方法：绘制对话框
    canvas.drawBitmap(bmpDialogBack, 0, DIALOG_START_Y, null); //绘制对话框背景
    alertMessage=alertMessage.replaceFirst("xx", city.getCityName());  //替换成实际的城市名称
/*将 alertMessage 字符串中的“xx”内容替换为实际的城市名称*/
    drawString(canvas, alertMessage);                                      //绘制提示文本
    Paint paint = new Paint();                                             //创建画笔
    paint.setTextSize(DIALOG_WORD_SIZE);                                   //设置字体大小
    paint.setAntiAlias(true);                                              //设置抗锯齿
    paint.setARGB(255, 42, 48, 103);                                       //设置字体颜色
/*分别在对话框中绘制“划拨”和“忽略”按钮*/
//绘制“划拨”按钮
    canvas.drawBitmap(bmpDialogButton, DIALOG_BTN_START_X, DIALOG_BTN_START_Y,
    null);                                                                 //绘制按钮图片
    canvas.drawText("划拨",                                                //绘制按钮上的文字
    DIALOG_BTN_START_X+DIALOG_BTN_WORD_LEFT,
    DIALOG_BTN_START_Y+DIALOG_WORD_SIZE+DIALOG_BTN_WORD_UP,
```

```
        paint);
        //绘制“忽略”按钮
        canvas.drawBitmap(bmpDialogButton, DIALOG_BTN_START_X+DIALOG_BTN_SPAN,
        DIALOG_ BTN_START_Y, null);
        canvas.drawText("忽略",                              //绘制按钮上的文字
            DIALOG_BTN_START_X+DIALOG_BTN_SPAN+DIALOG_BTN_WORD_LEFT,
            DIALOG_ BTN_START_Y+DIALOG_WORD_SIZE+DIALOG_BTN_WORD_UP,paint);
    }
```

（3）当玩家单击 FoodAlert 对话框上的按钮时，onTouch 方法将会对单击事件进行处理，该方法的代码如下。

```
    public boolean onTouch(View v, MotionEvent event){          //方法：处理单击屏幕事件
        if(event.getAction() == MotionEvent.ACTION_DOWN){       //事件为单击屏幕
            int x = (int)event.getX();                          //获取屏幕按下的 x 坐标
            int y = (int)event.getY();                          //获取屏幕按下的 y 坐标
    /*当玩家单击“划拨”按钮时的处理代码，由于单击“划拨”按钮需要弹出城池管理界面，所以需要
将游戏设置为响应的状态，同时将 GameView 中记录当前游戏提示的成员变量 currentGameAlert 赋值为
null，然后将监听器重新设置为 GameView*/
            if(x>DIALOG_BTN_START_X && x<DIALOG_BTN_START_X+DIALOG_BTN_WIDTH
            && y>DIALOG_BTN_START_Y && y<DIALOG_BTN_START_Y+
            DIALOG_BTN_ HEIGHT){                                //单击的是“确定”按钮
                    gameView.setStatus(97);                     //弹出自己城池的管理界面
                    gameView.setCurrentGameAlert(null);
                    gameView.setOnTouchListener(gameView);
    /*当玩家单击“忽略”按钮时的处理代码，单击“忽略”按钮后执行的代码主要职责是恢复游戏
状态*/
            }else if(x>DIALOG_BTN_START_X+DIALOG_BTN_SPAN && x<DIALOG_BTN_
            START_X+DIALOG_BTN_SPAN+DIALOG_BTN_WIDTH && y>DIALOG_BTN_
            START_Y && y< DIALOG_BTN_START_Y+ DIALOG_BTN_HEIGHT){
            //单击“忽略”按钮
                gameView.setCurrentGameAlert(null);
                gameView.setStatus(0);                          //设置游戏状态为待命
                gameView.setOnTouchListener(gameView);          //监听器设置为 GameView
            }}
            return true;
        }
```

☑ 代码第 5～10 行为当玩家单击“划拨”按钮时的处理代码，由于单击“划拨”按钮需要弹出城池管理界面，所以需要将游戏设置为响应的状态，同时将 GameView 中记录当前游戏提示的成员变量 currentGameAlert 赋值为 null，然后将监听器重新设置为 GameView。

☑ 代码第 11～18 行为当玩家单击“忽略”按钮时的处理代码，单击“忽略”按钮后执行的代码主要职责是恢复游戏状态。



任务 39　HeroBackDataThread 中对 FoodAlert 的调用

【任务情境】

本书在前面对 HeroBackDataThread 做过简单的介绍，该线程主要负责定时修改英雄的一些后台数据，并根据一定规则产生粮草危机、敌军来袭等事件，这些事件的发生都需要用到 GameAlert。

【相关知识】

下面就以 FoodAlert 为例来说明 GameAlert 在游戏中的用法。

在 HeroBackDataThread 的 run 方法中，对于城池粮草的处理是按照如下流程来进行的。

- ☑ 遍历英雄拥有的城池，对每个城池的粮草进行衰减。
- ☑ 检查衰减后剩余的粮食，是否小于最小值。如果小于最小值，则判断这是该城池的第几次粮草警报，如果是第二次，则新建一个 PlainAlert 通知玩家该城池已不攻自破。
- ☑ 如果这是该城池的第一次粮草警报，则创建一个 FoodAlert 对象提示玩家是否进行处理，并将该城池记录下来。

下面来看 run 方法中对上述逻辑的具体实现，其代码如下。

```
public void run(){                    //HeroBackDataThread 线程的 run 方法
    while(flag){                      //粮食衰减
        foodCount++;                  //对计数器自加操作，当计数器满 5 次时才对城池的粮草进行衰减
        if(foodCount == 5){           //满足条件就进行粮食衰减
            foodCount = 0;
            hero.setFood((int)(hero.getFood()*0.95));      //按比例减少英雄的粮食
            ArrayList<CityDrawable> cityList = hero.getCityList();
/*遍历英雄的城池列表对每座城池的粮草进行衰减的代码*/
            for(CityDrawable city:cityList){         //遍历英雄城池列表，减少各城池的粮食
                city.food -= (int)(city.food*0.15f);
                if(city.food <= MIN_FOOD){           //如果城中粮草小于某个值就报警
                    if(ignoredCity == city){         //如果之前已经警报过了，那城池就分崩离析吧
                        GameView gv = hero.father;
                        city.setBackToInit();   //恢复城市的信息到默认
                        String alertMessage="由于你没有及时输送粮草，"+city.getCityName()+"
                        由于粮草危机已经" +"不攻自破，将领和老百姓已经归顺了
                        "+COUNTRY_ NAME[city.getCountry()]+"，不再属于你了。";
                        PlainAlert pa = new PlainAlert(gv, alertMessage,
/*创建了一个 PlainAlert 对象来提示玩家该城池已经失去*/
//创建 PlainAlert 对象
```

```
GameView.dialogBack, GameView.dialogButton);
                                        hero.father.setCurrentGameAlert(pa);
/*将创建的 GameAlert 对象引用赋值给 GameView 中用于记录当前 GameAlert 对象的成员遍历*/
//设置为当前提示
                                        emptyCity = city;            //记录失去的城池
                                    }
                                    else{                            //如果是第一次出现粮草危机
                                        FoodAlert fa = new FoodAlert(hero.father, city, GameView. dialog
Back, GameView.dialogButton);
/*创建了一个 FoodAlert 对象来提示玩家对粮草告急城池进行处理*/
                                        hero.father.setCurrentGameAlert(fa);
/*将创建的 GameAlert 对象引用赋值给 GameView 中用于记录当前 GameAlert 对象的成员遍历*/
                                        ignoredCity = city;          //将第一次粮草危机的城池记录下来
                                    }}}
/*将已经有过两次粮草警报的城池从英雄拥有的城池列表中移除，同时将该城池添加到用于存放英雄
城池之外的城池列表中*/
                            if(emptyCity != null){                   //删掉因粮草危机而造成的空城
                                hero.cityList.remove(emptyCity);
                                hero.father.allCityDrawable.add(emptyCity);
                                emptyCity = null;
                            }}
                        //科研制造
                        …//此处省略 run 方法的其他逻辑代码
                        try{Thread.sleep(sleepSpan);}                //休眠一段时间
                        catch(Exception e){e.printStackTrace();}     //打印捕获异常
                    }}
```

【项目小结】

本项目介绍的游戏提示模块的开发，其中涉及到的类主要有 GameAlert 及其子类 FoodAlert、WarAlert、PlainAlert 和 GameOverAlert，其作用是游戏中发生粮草危机、敌军突袭等事件时需要提示玩家进行相应处理。希望读者能够用心学习，熟练掌握。

小知识二十：2012 年值得移动开发者关注的变化及趋势

每年都会有不同的人对于下一年某个产业的发展趋势做出预测，关注移动互联网多年，同时在这个行业里也做了一些事情，结合最近的一些思考，下面简单谈一下我对 2012 年移动互联网在中国发展趋势的一点看法。每当有人做下一年预测的时候，通常会列出 10 个趋势，从我目前的理解来看，只能看到 5 个方面，简单和大家交流一下。

1. App Store 中国区收入激增

我们此前听过了太多关于通过 App Store 创富的神话，但那都是在海外市场。2011 年，随着苹果成功地推出了人民币支付的解决方案，可以相信，2012 年 App Store 中国区的收



入将激增。对中国市场来讲，在面对猖獗的盗版情况下，便捷的支付方式则尤为重要。尽管目前 App Store 的支付方案对于中国用户来讲还有一定的门槛，但毕竟苹果开始重视中国 App Store 这块市场，这是好事。

2. Windows Phone 7 平台值得关注

目前，国内的开发者几乎把所有精力全部投入到了 iOS 和 Android 两个平台之上，鲜有独立的开发者关注 Windows Phone 7 这个平台。微软与诺基亚的战略合作，HTC、LG、三星、华为、中兴等手机厂商以及宏基、华硕等传统硬件厂商的加入对于 Windows Phone 7 的推广和普及都将起到重要的作用。在海外，我们看到已有大量开发者在该平台进行投入，Windows Phone 7 的应用程序已经突破 5 万个。在国内，腾讯、微软、百度等公司也均已推出该平台的产品并加快该领域的人才储备。在接下来的一年，Windows Phone 7 平台绝对值得开发者和投资人关注。

3. 线下商户拥抱移动互联网

随着电子商务的迅猛发展，很多线下的商户或多或少受到了冲击。对于线下商户来讲，进店率是最为重要的指标之一。手机的随身携带性结合手机自带的定位、信息推送等特性，使其成为线下商户进行营销推广的最好渠道。这类产品和服务的进入门槛看似很低，但是真正执行起来非常困难。看似简单但解决困难的问题就有两个：如何给商户和用户同时提供用户体验优良的产品？如何获取商户和用户？这只是最基础的两个问题，在解决这两个问题的前提下，打通产业链，这个领域一定会有大公司出现。

4. 移动社区突飞猛进

腾讯太强大了，社区还有机会吗？我的回答是有，并且机会还非常大。腾讯沉积在 QQ IM 上的关系链决定了它可以轻松地通过关系导入并建立一个强大的社区，但也决定了它不可能在任何社区类型上都可以通过导入关系链成功。在 PC 互联网上，我们看到了人人网、微博都甩开了腾讯，成为非常大的社区平台。转向移动互联网时代，深度利用手机特性，同时能建立独特数据或者独特人际关系的社区将取得迅猛发展。

5. 儿童教育市场将成 App Store 重要细分市场

在 iPhone 推出之后，海外有多家公司推出了面向儿童的教育类产品，或许由于 iPhone 屏幕尺寸等问题，这个市场直到 iPad 推出之后才获得快速发展。2011 年苹果第四季度财报显示，苹果当季共销售出 1112 万台 iPad 设备，而截止到 2011 年 10 月，苹果累计销售了 4000 万台 iPad 设备。随着 iPad 的普及以及父母对于儿童教育观念的转变，基于移动设备，尤其是平板设备的儿童教育类产品或将受到父母的追捧。目前，国内已经有多家公司推出了基于 iOS 设备的有声阅读类产品，这类产品相对来讲比较简单，而这只是基于 iPad 儿童教育类产品的一个代表，而更互动、更创新的儿童教育类产品值得期待。

综合实训十四　微博随身之相册管理模块的开发

【问题情境】

下面将介绍相册管理模块的开发，相册管理模块包括查看相册列表、修改相册权限和查看相册照片等功能，由 MyAlbumListActivity 和 AlbumActivity 来实现。

【拓展知识】

1. 相册查看和修改功能的实现

查看个人相册和修改相册权限的功能是由 MyAlbumListActivity 实现的，同 MyDiary Activity 类似，MyAlbumListActivity 通过 ListView 将个人相册名称及“查看”和“修改权限”显示到屏幕上。MyAlbumListActivity 类的代码如下。

```
package wyf.wpf;                              //声明包语句
import java.util.ArrayList;                   //引入相关类
...//此处省略部分引入相关类的代码
import android.widget.Toast;
public class MyAlbumListActivity extends Activity{
    MyConnector mc = null;                    //声明 MyConnector 对象
    ListView lvAlbumList = null;              //ListView 对象的引用
    List<String []> albumInfoList = null;     //存放相册信息的 List
    String albumInfoArray [] = null;          //存放相册信息的
    String uno = null;                        //存放用户的 ID
    int newAccess = -1;                       //记录新设置的权限
    String [] accessOptions={"公开","好友可见","仅个人可见"};
    int albumIndexToChange = -1;              //记录要更改权限的相册在信息列表中的索引
    String albumToChange = null;              //记录要更改权限的相册 ID
    String accessToChange = null;             //记录要更改的权限
    BaseAdapter ba = new BaseAdapter() { //此处省略创建 BaseAdapter 对象的代码};
    View.OnClickListener listenerToDetail = new View.OnClickListener() {
    //单击查看按钮后触发的监听器
        public void onClick(View v) {         //重写 onClick 方法
            Intent intent = new Intent (MyAlbumListActivity.this,AlbumActivity.class);
            //创建 Intent 对象
        intent.putExtra("uno", uno);                          //将用户 ID 添加到 Extra
        intent.putExtra("albumlist", albumInfoArray);         //相册列表信息添加到 Extra
        intent.putExtra("xid", albumInfoList.get(v.getId())[0]);
        //将要查看相册在 ListView 的位置添加到 Extra
        intent.putExtra("position", v.getId());               //将要查看相册的 ID 添加到 Extra
        intent.putExtra("from", 0);                           //设置 Extra 中的 from 的值
        startActivity(intent);                                //启动 AlbumActivity
        finish();                                             //结束本 Activity 的执行
    }
```

```
28  };
29  View.OnClickListener listenerToAcess = new View.OnClickListener() {
    //“修改相册权限”按钮监听器
30      public void onClick(View v) {                          //重写 onClick 方法
31          albumIndexToChange = v.getId();                    //获得被选中的相册索引
32          albumToChange = albumInfoList.get(v.getId())[0];   //获得被选中的相册 ID
33          newAccess = Integer.valueOf(albumInfoList.get(v.getId())[2]); //获得该相册当前权限
34          showMyDialog();                                    //自定义的显示对话框方法
35      }
36  };
37  Handler myHandler = new Handler(){              //此处省略 myHandler 对象的创建代码};
38  protected void onCreate(Bundle savedInstanceState) {   //重写 onCreate 方法
39      super.onCreate(savedInstanceState);
40      setContentView(R.layout.album_list);        //设置当前屏幕
41      Intent intent = getIntent();                //获得启动该 Activity 的 Intent
42      uno = intent.getStringExtra("uno");         //获得 Intent 中的 uno 的值
43      lvAlbumList = (ListView)findViewById(R.id.lvAlbumList);   //获得 ListView 对象
44      getAlbumList();                             //获得指定用户的相册列表
45  }
46  public void getAlbumList(){                     //方法：获得用户的相册列表}
47  public void showMyDialog(){                     //方法：显示修改权限对话框}
48  public int changeAlbumAccess(){                 //方法：修改相册权限}
49  protected void onDestroy() {                    //重写 onDestroy 方法}
50 }
```

☑ 第 8 行和第 9 行分别声明了用于存放相册列表信息的成员变量，这两个成员变量的不同在于 albumInfoList 把每个相册信息以一维数组的形式存放到 ArrayList 中；而 albumInfoArray 把每个相册信息以字符串的形式存放到一维数组中；albumInfoArray 用于将相册信息在不同的 Activity 之间传递。

☑ 第 12 行声明了一个 String 数组，该数组中的元素将作为修改权限的单选按钮对话框中的选项。

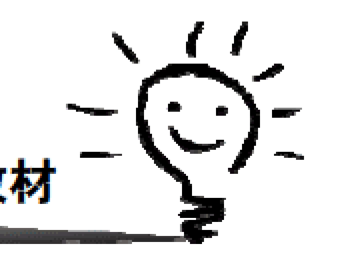
☑ 第 16 行创建了一个 BaseAdapter 对象，由于创建 BaseAdapter 对象的代码比较类似，主要是对 getView 方法的重写有所差别，故本书将不列出其具体代码。

☑ 第 17～28 行和 29～36 行创建了两个 OnClickListener 监听器 listenerToDetail 和 listenerToAccess，显示相册列表的 ListView 中所有的“查看”按钮将添加 listenerToDetail 监听器，而所有“修改权限”按钮将添加 listenerToAccess 监听器。

☑ 第 37 行创建了一个 Handler 对象，该对象与前面介绍过的 Handler 对象的功能类似，都是负责接收消息并调用 ListView 的 setAdapter 方法。

☑ 第 38～45 行为重写的 onCreate 方法，该方法的主要功能是读取启动该 Activity 的 Intent 中的值并设置当前显示的屏幕。然后调用 getAlbumList 方法获得用户的相册列表。

上述代码第 46 行省略了 getAlbumList 方法的代码，该方法的功能是连接服务器获取用户的相册列表，其代码如下。



```
1   public void getAlbumList(){                              //方法：获取个人相册列表
2       new Thread(){                                        //创建线程
3           public void run(){                               //重写的 run 方法
4               Looper.prepare();
5               try{
6                   if(mc == null){                          //检查 MyConnector 对象是否为空
7                       mc = new MyConnector(SERVER_ADDRESS, SERVER_PORT);
                        //创建 MyConnector 对象
8                   }
9                   mc.dout.writeUTF("<#GET_ALBUM_LIST#>"+uno);
                    //发出获取相册列表请求
10                  String reply = mc.din.readUTF();         //读取相册列表
11                  if(reply.equals("<#NO_ALBUM#>")){        //判断相册列表是否为空
12                      Toast.makeText(MyAlbumListActivity.this, "您还没有上传过照片",
                        Toast.LENGTH_LONG).show();           //输出提示消息
13                      Looper.loop();
14                      return;
15                  }
16                  albumInfoArray = reply.split("\\$");     //切割字符串
17                  albumInfoList = new ArrayList<String []>();//创建存放相册信息的 ArrayList
18                  for(String s:albumInfoArray){
19                      String [] sa = s.split("\\|");       //切割字符串
20                  albumInfoList.add(sa);                   //将相册信息添加到 ArrayList 中
21                  }
22                  myHandler.sendEmptyMessage(0);           //发出 Handler 消息
23              }catch(Exception e){e.printStackTrace();}    //捕获并打印异常
24          }
25      }.start();
26  }
```

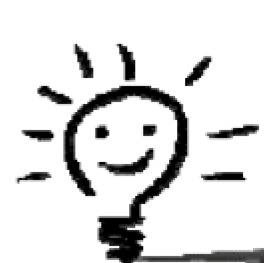
- ☑ 第 9 行将用户的 ID 和消息头进行组装生成请求消息并发送到服务端。第 10 行代码接收服务器返回的消息。
- ☑ 第 11～15 行判断接收到的服务器返回消息是否为“<#NO_ALBUM#>”，如果是，则通过 Toast 提示该用户还没有上传过照片。
- ☑ 第 16～22 行为当服务器返回的消息部位“<#NO_ALBUM#>”时执行的代码，首先创建一个用于存储相册信息的 ArrayList，然后对收到的字符串进行切割，提取出信息并添加到 albumInfoList 列表中。

2. 相片查看功能的实现

下面将介绍相片查看功能的开发。该功能由 AlbumActivity 实现。

AlbumActivity 中采用 Gallery 和 ImageSwitcher 显示图片，其中 Gallery 负责以缩略图的形式浏览所有相片，ImageSwitcher 负责显示指定的某张相片。同时 AlbumActivity 界面还包括一个“返回”按钮和一个只有在查看自己相册时才显示的“删除相片”按钮，AlbumActivity 的代码如下。

```
package wyf.wpf;                                             //声明包语句
```



```
import java.util.ArrayList;                                   //引入相关类
…//此处省略部分引入相关类的代码
import android.widget.TextView;
public class AlbumActivity extends Activity implements ViewSwitcher.ViewFactory{
    List<String []> photoInfoList = new ArrayList<String []>();    //创建存放照片信息的 List
    Bitmap [] photoList;                                      //存放图片的数组
    Gallery gl = null;                                        //Gallery 对象的引用
    ImageSwitcher is = null;                                  //ImageSwitcher 对象的引用
    Spinner sp = null;                                        //Spinner 对象的引用
    MyConnector mc = null;                                    //MyConnector 对象的引用
    String xid = "";                                          //存放相册的 ID
    String uno = "";                                          //存放用户 ID
    String visitor = "";                                      //存放访问者的 ID
    String pid = "";                                          //存放当前显示的照片 ID
    int from = -1;     //启动该 Activity 的来源，0：MyAlbumListActivity，1：AlbumListActivity
    List<String []> albumInfoList = new ArrayList<String []>();//存放相册信息、ID 和相册名称
    BaseAdapter baSpinner = new BaseAdapter(){//此处省略 Spinner 的 Adapter 的创建代码};
    BaseAdapter baGallery= new BaseAdapter(){//此处省略 Gallery 的 Adapter 的创建代码};
    OnItemClickListener myListener = new OnItemClickListener()
        {//此处省略 Gallery 的监听器的创建代码};
    Handler myHandler = new Handler(){                        //此处省略 Handler 对象的创建};
    protected void onCreate(Bundle savedInstanceState) {      //重写 onCreate 方法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.album);                       //设置当前屏幕
        Intent intent = getIntent();                          //获取启动该 Activity 的 Intent
        uno = intent.getStringExtra("uno");                   //获得 Extra 字段的 uno
        visitor = intent.getStringExtra("visitor");
        from = intent.getIntExtra("from", -1);                //获得 Extra 字段的 from
        int position = intent.getIntExtra("position", 0);     //获得被选中的相册
        String [] albumInfoArray = intent.getStringArrayExtra("albumlist");//获得相册信息数组
        xid = intent.getStringExtra("xid");                   //获得被选中的相册编号
        albumInfoList = new ArrayList<String []>();
        for(String s:albumInfoArray){                         //遍历信息数组
            String [] sa = s.split("\\|");
            albumInfoList.add(sa);                            //构建相册信息列表
        }
        sp = (Spinner)findViewById(R.id.spAlbum);             //获得 Spinner 对象
        sp.setAdapter(baSpinner);                             //设置 Spinner 对象的 Adapter
        sp.setSelection(position);                  //选中在前一个 Activity 中被选中的相册
        sp.setOnItemSelectedListener(new OnItemSelectedListener() { //为 Spinner 添加监听器
        gl = (Gallery)findViewById(R.id.galleryPhoto);              //获得 Gallery 对象
        gl.setOnItemClickListener(myListener);      //设置 Gallery 的 OnItemClickListener 监听器
        is = (ImageSwitcher)findViewById(R.id.isPhoto);       //获得 ImageSwitcher 对象
        is.setFactory(this);//设置 ImageSwitcher 的 Factory
        is.setInAnimation(AnimationUtils.loadAnimation(this, android.R.anim.fade_in));
        //设置 ImageSwitcher 的 In 动画
        is.setOutAnimation(AnimationUtils.loadAnimation(this,android.R.anim.fade_out));
        //设置 ImageSwitcher 的 Out 动画
        Button btnBack = (Button)findViewById(R.id.btnAlbumBack);//获得返回按钮 btnBack
```

```
48          btnBack.setOnClickListener(new View.OnClickListener() {});
49          Button btnDeletePhoto = (Button)findViewById(R.id.btnDeletePhoto);
50                if(visitor != null){                        //不是查看自己的相册
51                      btnDeletePhoto.setVisibility(View.GONE); //如果不是自己的相册，隐藏删除按钮
52          }
53          btnDeletePhoto.setOnClickListener(new
54    }
55    View.OnClickListener() {
56          public void getPhotoList(){//方法：获取指定相册的照片列表}
57          public void deletePhoto(){//方法：删除指定照片}
58          public View makeView() {//方法：重写 ViewSwitcher.ViewFactory 接口的 makeView 方法}
59    }
```

☑ 本程序中用户管理自己的相片和查看其他用户相册照片时都是通过 Album Activity 实现的，因此在 AlbumActivity 的代码中会判断当前是查看自己相册还是他人相册的代码。简单地说，如果启动 AlbumActivity 时在 Intent 对象中设置了名为“visitor”的 Extra 字段，则说明是在浏览他人的相册照片。

☑ 第 18 行和第 19 行分别创建了供相册下拉列表 Spinner 和相册照片浏览的 Galley 使用的 BaseAdapter 对象；第 20 行为 Gallery 对象创建了 OnItemClickListener 监听器。由于篇幅所限，将不再详述其代码。

☑ 第 25 行代码获得启动该 Activity 的 Intent 对象，第 26～36 行为读取 Intent 对象各个 Extra 值的代码，其中第 28 行的 from 成员记录启动该 Activity 的来源，用户单击“返回”按钮后将根据该成员变量的值返回到相应的 Activity。

☑ 第 40 行代码为 Spinner 对象添加 OnItemClickListener 监听器，当用户选择了下拉列表中的某个选项后，将会调用 getPhotoList 重新获得指定相册的照片列表。

☑ 第 43 行代码获得了 ImageSwitcher 对象的引用，第 44 行为 ImageSwitcher 添加了 ViewFactory 接口实现，第 45 行设置了 ImageSwitcher 切换图片时的 In 和 Out 效果。代码第 58 行为对接口 ViewSwitcher.ViewFactory 中 makeView 方法的重写。

到现在为止，本书已经对微博随身 Android 端的主要功能模块进行了简单介绍。由于篇幅有限，访问博友和搜索博友模块的具体代码并没有详细列出，这两个模块的实现方式与已介绍过的模块比较类似，读者可自行练习。

小知识二十一：谷歌、苹果亦敌亦友——激烈对抗、被迫合作

导语：美国《圣何塞信使报》网络版上发表署名克里斯·奥布莱恩（Chris O’Brien）的文章称，虽然谷歌和苹果在很多领域相互竞争，但二者仍然要被迫展开很多合作。这种亦敌亦友的状态还将持续一段时间。但随着苹果实力的日益壮大，这种关系随时都有可能破裂。到那时，就将引发硅谷有史以来最惊心动魄的分手事件。

1. 热核战争

谷歌和苹果的关系最近几年发生了戏剧性的变化，从亲密无间的兄弟变成了不共戴天的仇人。谷歌推出 Android 操作系统让史蒂夫·乔布斯（Steve Jobs）恨之入骨，这一点也在他的传记中得以体现。

“如果有必要，我会用尽最后一口气，我还将花光苹果 400 亿美元现金来纠正这一错误。我会毁掉 Android，因为这是一款偷来的产品。我想发动一场热核战争。”他说。

但这两大硅谷巨头之间的关系比这要复杂得多。目前为止，他们仍然彼此依赖，换句话说，“热核战争”不在考虑之列。

不过，最近有一些迹象显示，苹果可能会努力疏远谷歌。这也引发了我的好奇：谷歌将因此面临多大的风险？

事情其实并不算大：在 iPad 的一次发布会上，苹果高管放弃了谷歌地图，转而使用一款开放地图产品。这算不上什么杀手锏，但的确引发了外界对苹果与谷歌关系的猜测。

麦格理证券分析师本 • 沙赫特（Ben Schachter）在研究报告中写道：“如果苹果采取更激进的策略怎么办？”换句话说，如果苹果不再将谷歌作为 Safari 浏览器的默认搜索引擎，情况将会如何？

“如果苹果继续主导平板电脑，对谷歌的长期影响就将更为巨大。”沙赫特写道。

2. 关系复杂

很多用户都认为，苹果之所以使用谷歌搜索引擎，是因为该产品最优秀。尽管双方都不愿谈论这一话题，但谷歌为了成为苹果浏览器的默认搜索引擎，的确支付了大笔费用。

事实上，谷歌 2011 年在类似的交易中向苹果、Mozilla 和 MySpace 支付了总额 15 亿美元的资金。沙赫特估计，其中有 10 亿美元花在苹果身上，但实际情况仍未可知。

美国证券交易委员会（SEC）曾经要求谷歌公布这类交易的信息，以及具体盈利状况，但谷歌却成功地回避了这一要求。不过，谷歌还是承认，其中部分交易的确是在赔本经营。

有媒体报道称，美国联邦贸易委员会（FTC）已经将苹果列为谷歌反垄断调查的讯问对象。而双方的这一交易似乎正是关键所在。

苹果拒绝发表评论，谷歌也没有对沙赫特的测算以及该公司与苹果的关系进行回应。即使无法了解具体数额，沙赫特仍然认为，有一件事情是确定的：苹果产品对谷歌搜索的重要性正在日益加强。那么，在计算行业今后几年向移动设备转型的过程中，如果苹果在平板电脑市场的主导地位得以保持，该公司是否能够加强对谷歌的影响力？

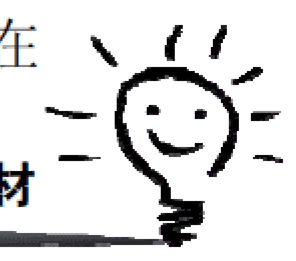
“如果苹果能够维持这种份额，谷歌就会面临麻烦。”沙赫特说，“苹果今后要问的是：你想给予你的直接竞争对手多少帮助？”

从这一点来讲，Android 的重要性就更加突出。虽然 Android 的普及率令人惊讶，但很多业内人士都批评该产品并未给谷歌带来任何贡献。但我们可以换个角度来考虑：如果苹果在智能手机市场获得与平板电脑相同的主导份额，谷歌想要成为默认的搜索服务提供商需要多花多少钱？沙赫特预计，单这一项每年就为谷歌节约了数亿美元。

3. 分手概率

然而，著名搜索行业分析师丹尼 • 沙利文（Danny Sullivan）却认为，苹果与谷歌之间的关系不会发生任何变化。作为科技博客 Search Engine Land 的创始人，沙利文已经关注搜索行业 15 年之久。在他看来，苹果取消谷歌默认搜索引擎地位的可能性微乎其微。

首先，沙利文认为，苹果自己没有开发搜索引擎。即使 Siri 吸引了很多关注，但用户体验仍然不够完美。而且，尽管微软对 Bing 投入了大量资源，并且有报道称，该公司正在



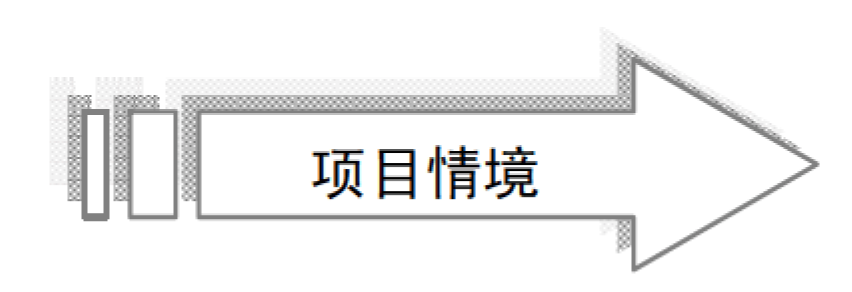
努力角逐苹果的默认搜索引擎地位，但用户认可度仍然不及谷歌。

“谷歌是一个十分强大的品牌，如果苹果转用 Bing，很多人仍会主动选择谷歌。”沙利文说，“人们做梦都不会想到这种事情。”

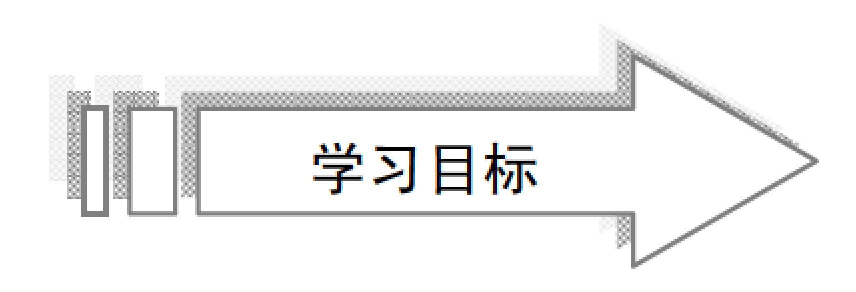
但沙利文表示，谷歌的确预见到这种问题，因此才开发了 Android 和 Chrome 浏览器这样的产品。这也导致谷歌与苹果和 Mozilla 等合作伙伴交恶，使得他们不太可能在反垄断调查中维护谷歌的利益。

如今，谷歌和苹果虽相互携手，但却各怀心事，甚至渐生敌意，但还不足以就此分手。如果苹果迈出第一步，那好戏就将上演，这无疑会成为硅谷有史以来最惊心动魄的分手事件。

项目十五
游戏后回味



到此为止，本游戏的开发已经基本完成，游戏中各个模块的功能也都得到实现。在每个游戏开发结束后，都要回头进行反思，思考游戏中是否有需要且可以优化和提升的地方。



- 学习如何全面考量和评价一个游戏，能够准确指出其优缺点及可改进之处
- 练习针对本节提出的问题对将本游戏继续优化和提升

任务 40　游戏的优化和改进

【任务情境】

“战国英雄传”游戏中仍然存在一些可以优化和提升的地方，现将这些可改进的地方总结如下。

【相关知识】

1. 丰富故事情节

本游戏将故事背景设在了战国时代，因此可以将历史上的重大事件融入到故事中，玩家可以通过对这些事件的不同反应来影响游戏的进程。

2. 多处存档

游戏中只提供了一个保存点，有兴趣的读者可以在此进行改进，增加几个保存点供玩家选择。这可提高游戏的合理性。

3. 公式的增强

游戏中使用的计算公式大都比较简单，比如技术攻击力以及防御力的公式，在真实的游戏开发中，此公式可能会非常复杂。

【项目小结】

本项目总结了本游戏可以优化和提升的部分，主要包括丰富故事情节、多出存档、公式的增强，在开发其他游戏甚至更多项目完成后，都需要对已完成的功能回头想一想，哪里需要进一步优化和改进。

综合实训十五　微博随身之游戏的优化和改进

【问题情境】

本实训对微博随身 Web 端和 Android 手机端的各个功能模块及实现方式进行了简单的说明。尽管如此，本系统中仍然存在一些可以提升和改进的地方，现将这些可优化的部分列出，读者朋友如果有兴趣，可以自己去实现。



【拓展知识】

1. Web 端的美工

本系统由于主要侧重于 Android 手机端功能的设计与实现，并没有在 Web 端下过多的工夫，因此相比其功能来说，Web 端美工效果可能要略微差一些，这一点可以适当优化。

2. Android 端个人资料修改

在微博随身的 Web 端，用户可以在个人资料页面修改自己的个人资料，Android 手机端也可以添加类似的功能。

3. 广告模块

可以在 Android 手机端和 Web 端添加广告模块，广告的显示位置可以是 Web 端的页面两侧或者 Android 手机端的登录界面等位置。广告显示的内容是从服务器获取，因此还需要添加一个广告管理模块对广告进行后台设置。

参 考 文 献

[1]吴亚峰，苏亚光．Android 2.0 游戏开发实战宝典[M]．北京：人民邮电出版社，2010．
[2]吴亚峰，索依娜．Android 核心技术与实例详解[M]．北京：人民邮电出版社，2010．
[3]杨丰盛．Android 应用开发揭秘[M]．北京：机械工业出版社，2010．